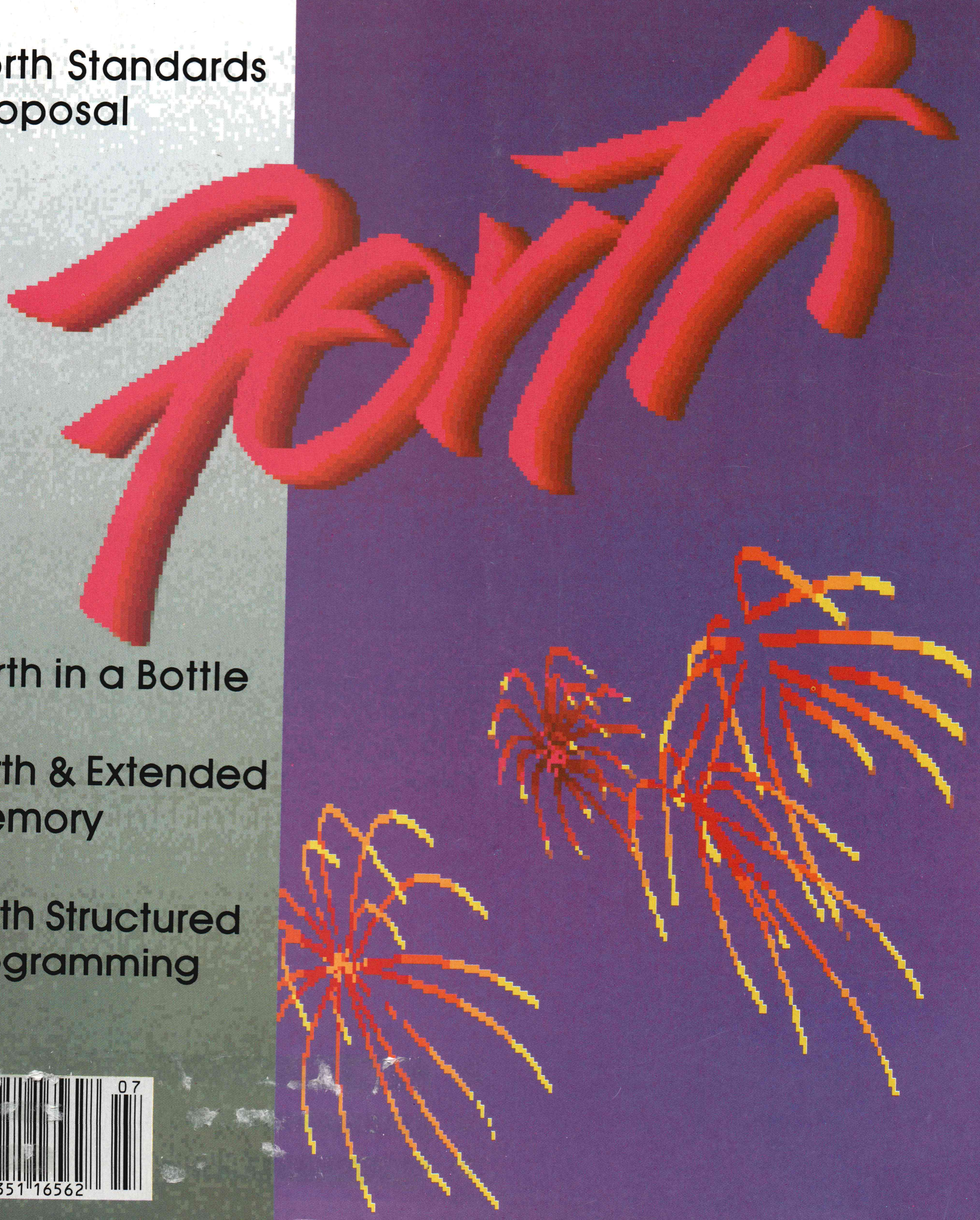
Dr. Dobb's Journal of

Software Tools

FOR THE PROFESSIONAL PROGRAMMER

#117 JULY 1986
2.95 (3.95 CANADA)

Forth Standards Proposal

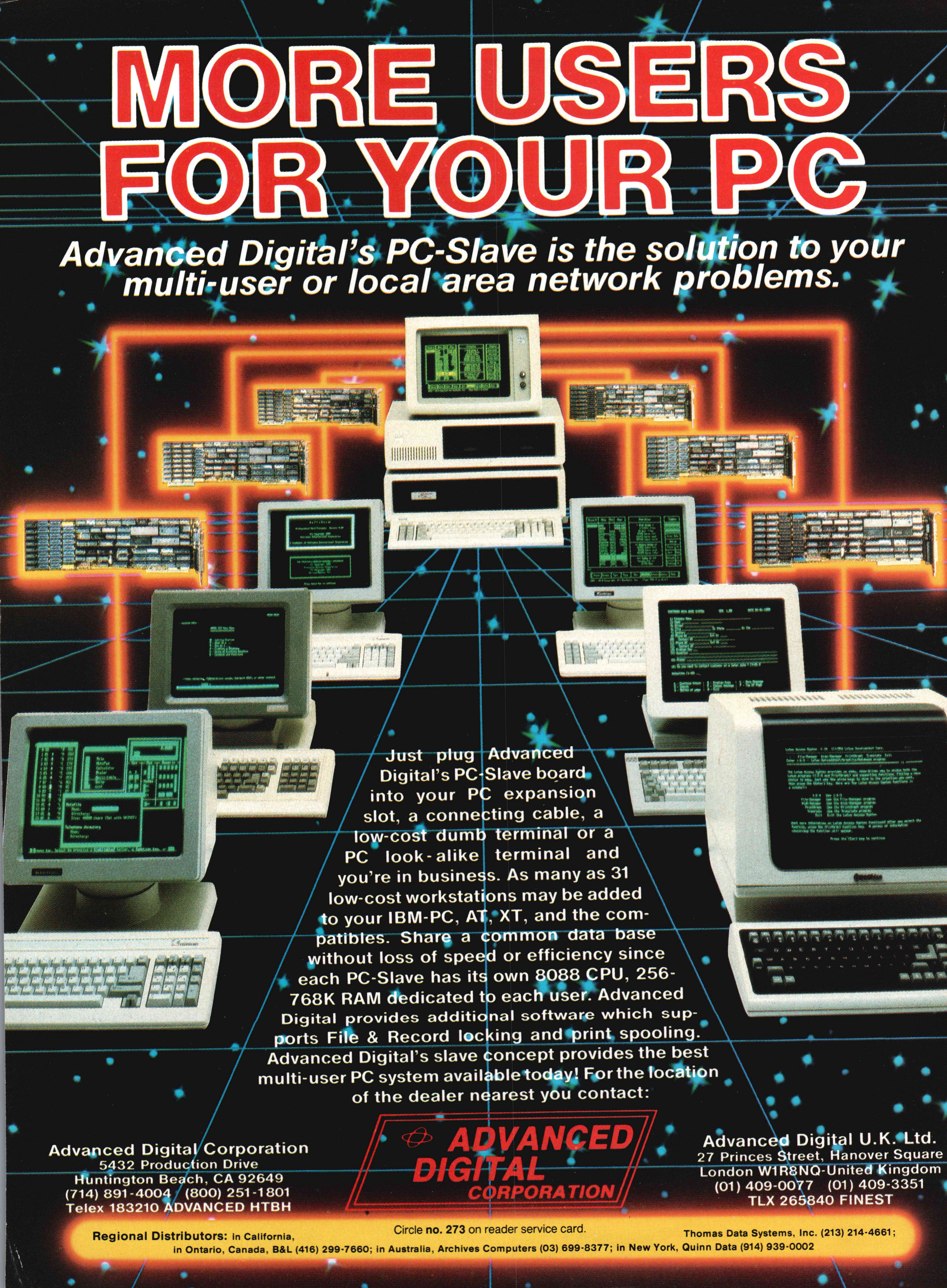


Forth in a Bottle

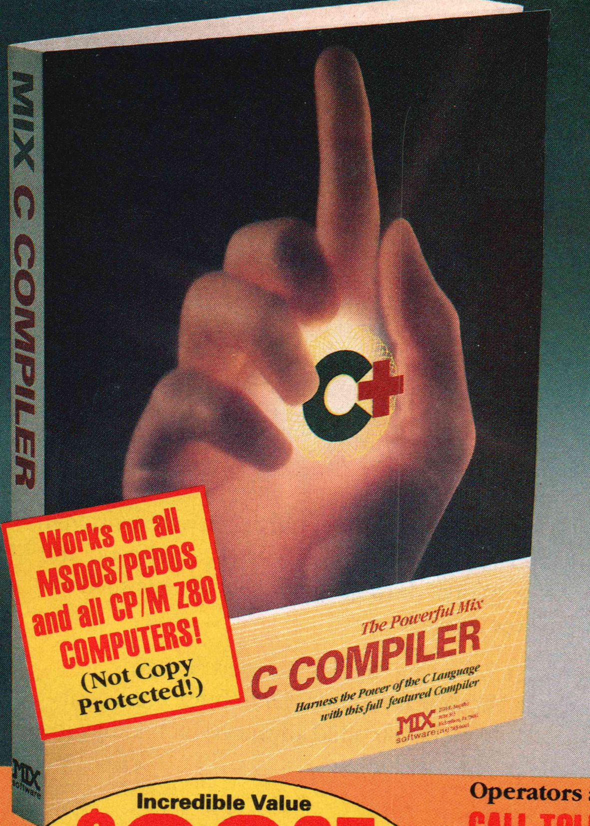
Forth & Extended Memory

Forth Structured Programming

0 38351 16562 07



MORE USERS FOR YOUR PC
Advanced Digital's PC-Slave is the solution to your multi-user or local area network problems.
Just plug Advanced Digital's PC-Slave board into your PC expansion slot, a connecting cable, a low-cost dumb terminal or a PC look-alike terminal and you're in business. As many as 31 low-cost workstations may be added to your IBM-PC, AT, XT, and the compatibles. Share a common data base without loss of speed or efficiency since each PC-Slave has its own 8088 CPU, 256-768K RAM dedicated to each user. Advanced Digital provides additional software which supports File & Record locking and print spooling. Advanced Digital's slave concept provides the best multi-user PC system available today! For the location of the dealer nearest you contact:
Advanced Digital Corporation
5432 Production Drive
Huntington Beach, CA 92649
(714) 891-4004 (800) 251-1801
Telex 183210 ADVANCED HTBH
ADVANCED DIGITAL CORPORATION
Advanced Digital U.K. Ltd.
27 Princes Street, Hanover Square
London W1R8NQ-United Kingdom
(01) 409-0077 (01) 409-3351
TLX 265840 FINEST
Circle no. 273 on reader service card.
Regional Distributors: in California, Thomas Data Systems, Inc. (213) 214-4661; in Ontario, Canada, B&L (416) 299-7660; in Australia, Archives Computers (03) 699-8377; in New York, Quinn Data (914) 939-0002



C FEVER *catch it!*

It's becoming an epidemic . . . everyone is switching to C! First there were a few hackers, then came the college students, next the major software houses, and now the rest of the programming world. Programmers everywhere are infected with the desire for SPEED, POWER, and PORTABILITY.

It's time to face the inevitable. You're going to catch the fever too! When you do, give us a call. We've got the best cure—an illustrated guide to the C language, plus a complete program development system. Everything you need to master the C programming language . . . all at a price that's less than the cost of a book!

But don't let this price fool you. Our system is powerful; it compiles twice as fast as the others, is completely standard, and it's very easy to use. Most C compilers are designed for wizards. We have designed ours for you!

What do you get for a mere $39.95?

- **A 450 Page book filled with sample programs, plus . . .**
- **A fast, standard, full featured C compiler** that supports all data types and the latest features like bit fields, enumerations, structure assignment, and passing/returning structures.
- **A fast linker** that loads separately compiled files, searches libraries, and builds an executable program.
- **An extensive library of more than 170 functions** (including the standard C functions and the computer specific functions that provide direct access to the operating system and BIOS).
- **Tools that allow you to optimize** your programs for minimal space or maximum speed.

Incredible Value AT ONLY $39.95 With 30 Day Money-Back Guarantee

Operators are standing by . . . Please use this Number for ORDERS ONLY!

CALL TOLL FREE FOR RUSH ORDER DELIVERY!

1-800-523-9520

IN TEXAS, PLEASE CALL TOLL FREE 1-800-622-4070

For Technical Support Please call 1-214-783-6001

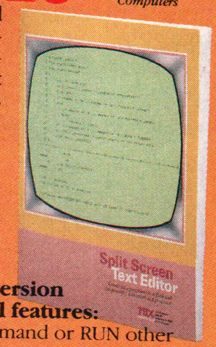
MIX Software, Inc. / 2116 E. Arapaho / Suite 363 / Richardson, Texas 75081

Or contact our Worldwide Distributors direct in:

Canada: Saraguay Software 1-416-923-1500
Australia: Techflow 047-586924
Switzerland: DMB Communication CH-1-825-53-29
France: Info/Tech 1-43-44-06-48

Split Screen Text Editor

an Incredible Value AT ONLY $29.95 *(Not Copy Protected!) Works on all MSDOS/PCDOS and CP/M Z80 Computers*



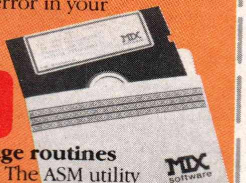
Our high powered editor is great for editing high level languages. **It works just like Micropro's Wordstar,**™ but macros allow you to create your own custom editor, and the split-screen feature lets you edit two files at the same time.

The MSDOS/PCDOS version is loaded with special features:

- Execute any DOS command or RUN other programs from the editor.
- Quickly edit files as large as 300,000 characters.
- Compile MIX C programs directly from memory. The editor automatically positions the cursor to the first error in your program.

ASM UTILITY

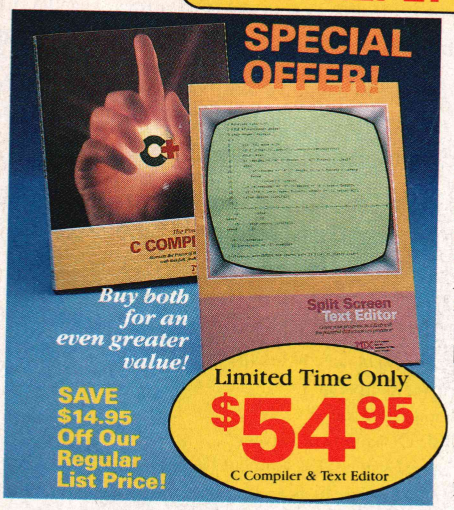
an Incredible Value AT ONLY $10



Call assembly language routines from your C programs. The ASM utility works with Microsoft's MASM or M80 assembler. Macros make it easy! Works just as if you were calling a C function, and you can even call C functions from assembly language. Lots of useful assembly language functions are included as examples.

RUSH REPLY ORDER FORM!

SPECIAL OFFER!



Buy both for an even greater value!

SAVE $14.95 Off Our Regular List Price!

Limited Time Only $54.95 C Compiler & Text Editor

Description	Quantity	PRICE	Total Order
Split-Screen Text Editor	______	$29.95	$______
C Compiler	______	$39.95	$______
C and Text Editor (Special)	______	$54.95	$______
ASM Utility	______	$10.00	$______
Texas Residents Add 6.125% Sales TAX			$
Shipping Charges (See at Right)			$
TOTAL OF YOUR ORDER:			$

Please check method of payment:
☐ Check ☐ Money Order ☐ MasterCard/VISA

Your Card #: ____________________

Expires ____________________

Shipping Charges: (No charge for ASM Utility)
In the U.S.A.: Add $5.00 per Order.
In CANADA: Add $10.00 per Order.
OVERSEAS: Add $10.00 per Text Editor. Add $20.00 per C Compiler. Add $30.00 for combined C Compiler and Text Editor.

Operating System: *(Check one)*
☐ CP/M Z80 ☐ MSDOS/PCDOS

Specify Your Computer Name ____________________

Specify Disk Format ____________________

NAME ____________________

Telephone A/C (______) ____________________

Street ____________________

City ____________________

State ____________________

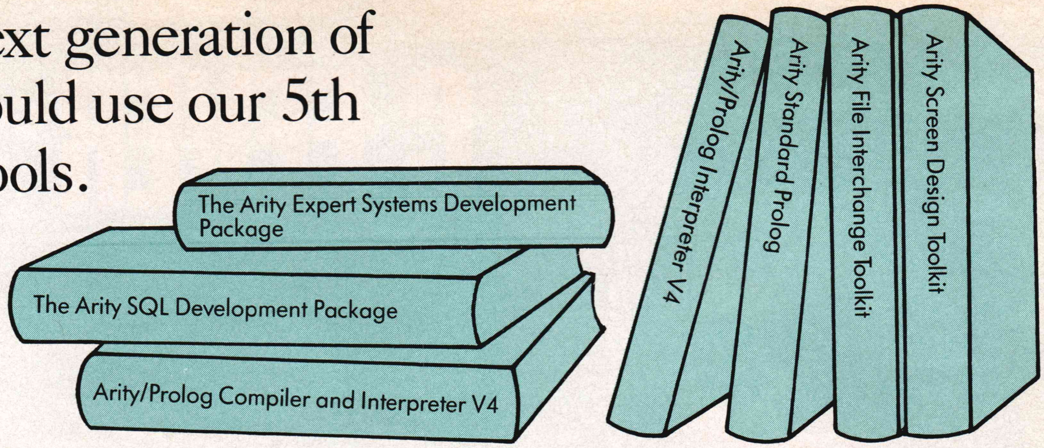
Country ____________ ZIP ____________

MIX software
2116 East Arapaho
Suite 363
Richardson, Texas, 75081

Ask about our Volume Discounts!
Call 1-214-783-6001

D

Why your next generation of products should use our 5th generation tools.



Arity's integrated family of programming tools allows you to combine software written in Arity/Prolog, the best of the fifth generation languages, with Arity SQL, the best of the fourth generation languages, and with conventional third generation languages such as C or assembly language to build your smarter application.

You can use Arity/Prolog to build expert systems using the Arity Expert Systems Development Package. Or to build natural language frontends. Or to build intelligent information management systems. Arity/Prolog lets you build advanced technology into your vertical applications package.

And more...

That's not the whole story. Arity's products are all designed to be fast, powerful, serious. Each of our products contains unexpected bonuses. Such as a one gigabyte virtual database integrated into Arity/Prolog. The most powerful of its kind on a PC.

Quality first. Then price.

In order to be the best, we had to prove it to our customers. Our tradition of quality software design is reflected in every product we sell. Quality first. Then price. And we always provide the best in customer support.

Our products are not copy protected. We do not charge royalties. We offer generous educational and quantity discounts. And we have a 30 day money back guarantee.

Try us to know that we keep our promise on commitment to quality and reliability. Try us by using our electronic bulletin board at 617-369-5622 or call us by telephone—you can reach us at 617-371-2422.

Or fill in this coupon. Whether you order today or not, let us send you full descriptions of our integrated family of Arity products.

arity

We design and distribute high quality, serious application software for the IBM PC, XT, AT and all MS-DOS compatibles.

Circle **no. 121** on reader service card.

Please complete this form to place your order and/or request detailed information.

		Quantity	Info only
Arity/Prolog Compiler and Interpreter V4	$795.00	______	___
Arity/Prolog Interpreter V4	$350.00	______	___
Arity Standard Prolog	$ 95.00	______	___
Arity SQL Development Package	$295.00	______	___
Arity Expert System Development Package	$295.00	______	___
Arity Screen Design Toolkit	$ 49.95	______	___
Arity File Interchange Toolkit	$ 49.95	______	___
TOTAL AMOUNT (MA residents add 5% sales tax) (These prices include shipping to all U.S. cities)		$______	

NAME______________________

SHIPPING ADDRESS______________________

CITY/STATE/ZIP______________________

TELEPHONE______________________

Payment: ☐ Check ☐ PO ☐ AMEX ☐ VISA ☐ MC

Card #______________________ Exp. date__________

Signature______________________

ARITY CORPORATION • 358 BAKER AVENUE • CONCORD, MA 01742

arity

CONTENTS

VOLUME 11, ISSUE 7

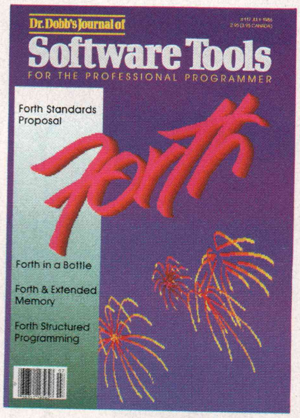
Dr. Dobb's Journal of
Software Tools

ARTICLES

COLUMNS

FORUM

PROGRAMMER'S SERVICES



About the Cover

The cover was designed by Janaia Donaldson and created by Dan Silva on a Commodore Amiga. Dan used the Deluxe Paint program he designed for Electronic Arts.

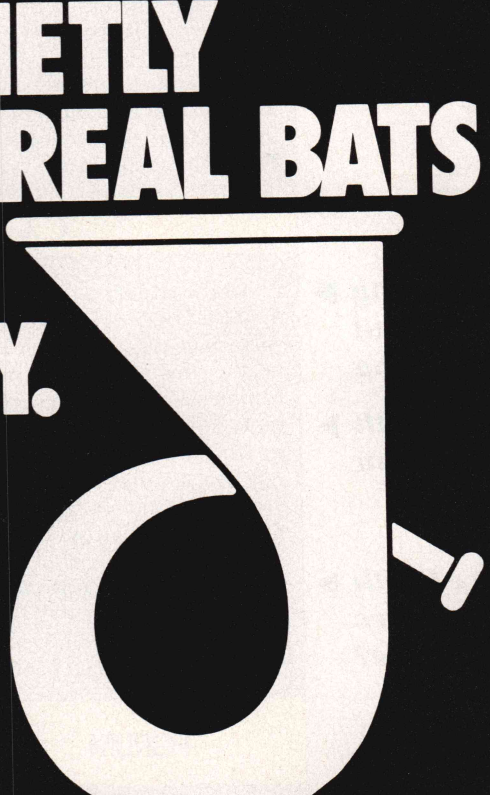
This Issue

Welcome to our sixth annual special issue on the Forth language. Our feature article proposes a comprehensive set of standards for extended control structures. Everett Carter puts Forth in a bottle, and Craig A. Lindley opens a window on easy-to-use implementations. We're also introducing a new columnist whose specialty is—you guessed it—Forth.

Next Issue

In the hot month of August why not dive into C? Last August, we broke ground with a new kind of software review. It was more technical and deeper in detail than the usual fare and used carefully designed "surgical" benchmarks. This year we reprise and improve the process with an up-to-date comparison of 17 C compilers for MS-DOS.

YOUR
COMPUTER LANGUAGE
IS QUIETLY
BREEDING REAL BATS
IN YOUR
BELFRY.



WE'RE OUT TO SAVE ONE MILLION FRUSTRATED PROGRAMMERS

You're on a roll, really pumped, writing the best code you have ever written and then—AAARGHHH!

Freeze dried in your tracks because the language you're using just won't let you achieve what you can conceive.

And you wanted to be a programmer.

So your choices are:

1) write around the problem by creating six pages of emetic code...

2) leave out that incredible idea that really puts your stamp of excellence on this program or...

3) get yourself a world class headache (or a stroke) by dropping into assembler.

Whatever you choose, by now you feel the language is out to get you—because it is.

Sure, no language is perfect, but you have to wonder, "Am I getting all I deserve?"

And, like money, you'll never have enough.

Pretty dismal, huh?

We thought so, too.

So we did something about it.

We call it CLARION.

You'll call it incredible.

With CLARION you can write, compile, run and debug complex applications in a New York afternoon.

Even if you're in Savannah.

It gives you the power and speed to create screens, windows and reports of such richness and clarity you would never attempt them with any other language.

Because YOU would have to write the code.

With CLARION you simply design the screens using our SCREENER utility and then CLARION writes the source code AND compiles it for you.

In seconds.

Likewise, you can use REPORTER to create reports. Remember, only CLARION can recompile and display a screen or report layout for modification.

And with no time wasted.

All the power and facilities you need to write great programs, faster than you ever dreamed of.

Programs that are easy to use.

Programs that are a pleasure to write.

And to you that means true satisfaction.

You've coveted those nifty pop-up help windows some major applications feature. But you can't afford the time and energy it takes to write them into your programs.

That's the way it used to be.

So we fixed that, too.

CLARION HELPER is an interactive utility that let's you design the most effective pop-up help screens that you can imagine. And they're "context sensitive," meaning you can have help for every field in your application.

Unlike the other micro languages, CLARION provides declarations, procedures and functions to process dates, strings, screens, reports, indexed files, DOS files and memory tables.

Imagine making source program changes with the CLARION EDITOR. A single keystroke terminates the EDITOR, loads the COMPILER, compiles the program, loads the PROCESSOR and executes the program. It's that easy!

Our data management capabilities are phenomenal. CLARION files permit any number of composite keys which are updated dynamically.

A file may have as many keys as it needs. Each key may be composed of any fields in any order. And key files are updated whenever the value of the key changes.

Like SCREENER and REPORTER, CLARION's FILER utility also has a piece of the CLARION COMPILER. To create a new file, you name the Source Module. Then you name the Statement Label of a file structure within it.

FILER will also automatically rebuild existing files to match a changed file structure. It creates a new record for every existing record, copying the existing fields and initializing new ones.

Sounds pretty complicated, huh?

Not with CLARION's documentation and on-line help screens. If you are currently competent in Basic, Pascal or "C" you can be writing CLARION applications in a day. In two days you won't believe the eloquence of your CLARION programs.

Okay, now for the best part of all. You can say it in CLARION for $295.00—complete. All you need is an IBM® PC, XT, AT or true compatible, with 320 KB of memory and a hard disk drive.

And we'll allow a full 30 day evaluation period. If you're not satisfied with CLARION, simply return it in its original condition for a full refund.

If you're not ready to take advantage of this no-risk opportunity, ask for our detailed 16 page color brochure. It vividly illustrates the elegance of CLARION. Consider it a preview of programming in the fast lane.

Either way, the call's a freebie.

SAY IT IN

CLARION™

DEPT. A1ST/2

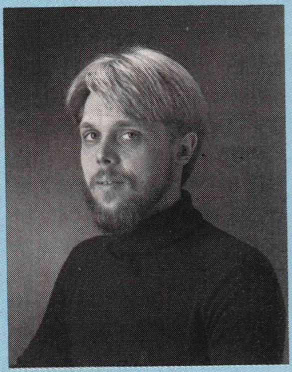
1-800-354-5444

BARRINGTON SYSTEMS, INC. • P.O. BOX 5580 • POMPANO BEACH, FL 33074 • 305/785-4555

IBM® is a registered trademark of International Business Machines Corporation. CLARION™ is a trademark of Barrington Systems, Inc. ©1986 Barrington Systems.

Circle no. 115 on reader service card.

EDITORIAL



Concern has been expressed that we are abandoning Forth. We're not. We have, though, devoted less attention to the language in the past year and intend to redress this, starting with this Forth issue. Michael Ham, our new columnist sharing the Structured Languages column, is an accomplished Forth programmer and one of the most eloquent voices in the Forth community. We're looking forward to publishing more Forth articles in future issues, particularly those that demonstrate the unique advantages of Forth's threaded, extensible, reverse Polish nature.

Michael Odawa, vice president of the Software Entrepreneur's Forum and director of the Software Services Association, sent us a letter about a new set of tax regulations that are up for consideration here in California. The new legislation, if passed, would be grossly unfair to independent software authors. Briefly, the proposed revision to the State Board of Equalization's Regulation No. 1502 would apply a sales tax of 6-7 percent (depending on the county) to all software royalty income, as well as some software-related business transactions (such as installation and maintenance of software). Neither of these forms of income is currently subject to sales tax, and neither should be—both represent charges for labor, not sales of physical assets. This ruling runs completely contrary to Sections 6011 and 6012 of the Revenue and Taxation code, in which labor charges are explicitly excluded from sales taxation.

One key issue here is the distinction between services and manufactured goods. Under the tax code, services are not subject to sales tax. The new legislation would make the medium used to deliver a new piece of software determine the taxability of the sale.

If you're a software author in California, this law would directly affect you. If you live elsewhere, you should be aware that wide-ranging legal precedents can be set by such legislation. Mr. Odawa can be reached through the Software Entrepreneur's Forum at (415) 854-7219. We urge programmers to educate California state legislators about this legislation.

Authors often ask, "How do I write a manuscript that you will want to publish?" Call me directly, tell me about your article, and make sure I send you copies of our *Writer's Guidelines* and *Style Sheet*. But that's only the first step. There are so many "good things to know" about writing for magazines that whole books have been written on the subject. Starting this month, I'd like to bring you some advice of my own. If you're working (or thinking of working) on an article, read on.

Keep in mind that space is often at a premium. Explain (briefly) what the problem was that you solved and explain the solution. Then summarize briefly and mention where your software is available.

Now is the time to get to work on articles for the next 68000 issue (January 1987). The deadline for that issue is September 15. Remember, if you get your article to me well before that date there will be time for rewrites—otherwise, if it's not perfect it might not be published. If you have any ideas you'd like to discuss, give me a call at (415) 366-3600.

Nick Turner
editor

Dr. Dobb's Journal of
Software Tools

Editorial
Editor-in-Chief *Michael Swaine*
Editor *Nick Turner*
Managing Editor *Vince Leone*
Assistant Editor *Sara Noah Ruddy*
Technical Editor *Allen Holub*
Contributing Editors *Ray Duncan*, *Allen Holub*, *Michael Ham*, *Namir Shammas*
Copy Editor *Rhoda Simmons*
Electronic Editor *Levi Thomas*

Production
Production Manager *Bob Wynne*
Art Director *Michael Hollister*
Assoc. Art Director *Alida Hinton*
Typesetter *Jean Aring*
Cover Artist *Janaia Donaldson*

Circulation
Newsstand Sales Mgr. *Stephanie Barber*
Circulation Director *Maureen Kaminski*
Book Marketing Mgr. *Jane Sharninghouse*
Circulation Assistant *Kathleen Shay*

Administration
Finance Manager *Treva Rafalski*
Business Manager *Betty Trickett*
Accounts Payable Supv. *Mayda Lopez-Quintana*
Accounts Payable Assts. *Denise Giannini*, *Kathy Robinson*
Accounts Receivable Mgr. *Laura Di Lazzaro*
Adm. Coordinator *Kobi Morgan*

Advertising Director
Robert Horton *(415) 366-3600*

Account Managers
Lisa Boudreau *(415) 366-3600*
Michele Beaty *(317) 875-8093*
Michael Wiener *(415) 366-3600*
Gary George *(404) 355-4128*
Promotions/Srvcs.Mgr. *Anna Kittleson*
Advertising Secretary *Michelle A. Davié*

M&T Publishing, Inc.
Chairman of the Board *Otmar Weber*
Director *C.F. von Quadt*
President and Publisher *Laird Foshay*

Dr. Dobb's Journal of Software Tools (USPS 307690) is published monthly by M&T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600, Second class postage paid at Redwood City and at additional entry points.

Article Submissions: Send manuscripts and disk (with article and listings) to the Assistant Editor.

Address correction requested: Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0888-3076**

Customer Service: For subscription problems call: outside CA 800-321-3333; within CA 619-485-9623 or 566-6947. For order problems call 415-366-3600.

Subscription Rates: $29.97 per year, U.S. Foreign rates: $56.97, air; $46.97 surface. Foreign subscriptions must be pre-paid in U.S. dollars, drawn on a U.S. Bank. For foreign subscriptions, TELEX: 752-351

Foreign Newsstand Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 6686-1520 TELEX: 620430 (WUI)

Entire contents copyright © 1986 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

People's Computer Company
Dr. Dobb's Journal of Software Tools is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit corporation.

AZTEC The C for Microcomputers

PC-DOS, MS-DOS, CP/M-86, Macintosh, Amiga, Apple II, CP/M-80, Radio Shack, Commodore, XENIX, ROM, and Cross Development systems

MS-DOS, PC-DOS, CP/M-86, XENIX, 8086/80x86 ROM

Manx Aztec C86

"A compiler that has many strengths . . . quite valuable for serious work"

Computer Language review, February 1985

Great Code: Manx Aztec C86 generates fast executing compact code. The benchmark results below are from a study conducted by Manx. The Dhrystone benchmark (CACM 10/84 27:10 p1018) measures performance for a systems software instruction mix. The results are without register variables. With register variables, Manx, Microsoft, and Mark Williams run proportionately faster, Lattice and Computer Innovations show no improvement.

	Execution Time	Code Size	Compile/ Link Time
Dhrystone Benchmark			
Manx Aztec C86 3.3	34 secs	5,760	93 secs
Microsoft C 3.0	34 secs	7,146	119 secs
Optimized C86 2.20J	53 secs	11,009	172 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Lattice 2.14	89 secs	20,404	117 secs

Great Features: Manx Aztec C86 is bundled with a powerful array of well documented productivity tools, library routines and features.

Optimized C compiler	Symbolic Debugger
AS86 Macro Assembler	LN86 Overlay Linker
80186/80286 Support	Librarian
8087/80287 Sensing Lib	Profiler
Extensive UNIX Library	DOS, Screen, & Graphics Lib
Large Memory Model	Intel Object Option
Z (vi) Source Editor -c	CP/M-86 Library -c
ROM Support Package -c	INTEL HEX Utility -c
Library Source Code -c	Mixed memory models -c
MAKE, DIFF, and GREP -c	Source Debugger -c
One year of updates -c	CP/M-86 Library -c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c are special features of the Aztec C86-c system.

Aztec C86-c Commercial System	**$499**
Aztec C86-d Developer's System	**$299**
Aztec C86-p Personal System	**$199**
Aztec C86-a Apprentice System	**$49**

All systems are upgradable by paying the difference in price plus $10.

Third Party Software: There are a number of high quality support packages for Manx Aztec C86 for screen management, graphics, database management, and software development.

C-tree $395	**Greenleaf $185**
PHACT $250	**PC-lint $98**
HALO $250	**Amber Windows $59**
PRE-C $395	**Windows for C $195**
WindScreen $149	**FirsTime $295**
SunScreen $99	**C Util Lib $185**
PANEL $295	**Plink-86 $395**

MACINTOSH, AMIGA, XENIX, CP/M-68K, 68k ROM

Manx Aztec C68k

"Library handling is very flexible . . . documentation is excellent . . . the shell a pleasure to work in . . . blows away the competition for pure compile speed . . . an excellent effort."

Computer Language review, April 1985

Aztec C68k is the most widely used commercial C compiler for the Macintosh. Its quality, performance, and completeness place Manx Aztec C68k in a position beyond comparison. It is available in several upgradable versions.

Optimized C	Creates Clickable Applications
Macro Assembler	Mouse Enhanced SHELL
Overlay Linker	Easy Access to Mac Toolbox
Resource Compiler	UNIX Library Functions
Debuggers	Terminal Emulator (Source)
Librarian	Clear Detailed Documentation
Source Editor	C-Stuff Library
MacRam Disk -c	UniTools (vi,make,diff,grep) -c
Library Source -c	One Year of Updates -c

Items marked -c are available only in the Manx Aztec C86-c system. Other features are in both the Aztec C86-d and Aztec C86-c systems.

Aztec C68k-c Commercial System	**$499**
Aztec C68d-d Developer's System	**$299**
Aztec C68k-p Personal System	**$199**
C-tree database (source)	**$399**
AMIGA, CP/M-68k, 68k UNIX	**call**

Apple II, Commodore, 65xx, 65C02 ROM

Manx Aztec C65

"The AZTEC C system is one of the finest software packages I have seen"

NIBBLE review, July 1984

A vast amount of business, consumer, and educational software is implemented in Manx Aztec C65. The quality and comprehensiveness of this system is competitive with 16 bit C systems. The system includes a full optimized C compiler, 6502 assembler, linkage editor, UNIX library, screen and graphics libraries, shell, and much more. The Apple II version runs under DOS 3.3, and ProDOS, Cross versions are available.

The Aztec C65-c/128 Commodore system runs under the C128 CP/M environment and generates programs for the C64, C128, and CP/M environments. Call for prices and availability of Apprentice, Personal and Developer versions for the Commodore 64 and 128 machines.

Aztec C65-c ProDOS & DOS 3.3	**$399**
Aztec C65-d Apple DOS 3.3	**$199**
Aztec C65-p Apple Personal system	**$99**
Aztec C65-a for learning C	**$49**
Aztec C65-c/128 C64, C128, CP/M	**$399**

Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized.

Manx Cross Development Systems

Cross developed programs are edited, compiled, assembled, and linked on one machine (the HOST) and transferred to another machine (the TARGET) for execution. This method is useful where the target machine is slower or more limited than the HOST, Manx cross compilers are used heavily to develop software for business, consumer, scientific, industrial, research, and educational applications.

HOSTS: VAX UNIX ($3000), PDP-11 UNIX ($2000), MS-DOS ($750), CP/M ($750), MACINTOSH ($750), CP/M-68k ($750), XENIX ($750).

TARGETS: MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x86 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

The first TARGET is included in the price of the HOST system. Additional TARGETS are $300 to $500 (non VAX) or $1000 (VAX).

Call Manx for information on cross development to the 68000, 65816, Amiga, C128, CP/M-68K, VRTX, and others.

CP/M, Radio Shack, 8080/8085/Z80 ROM

Manx Aztec CII

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."

80-Micro, December, 1984, John B. Harrell III

Aztec C II-c (CP/M & ROM)	**$349**
Aztec C II-d (CP/M)	**$199**
C-tree database (source)	**$399**
Aztec C80-c (TRS-80 3 & 4)	**$299**
Aztec C80-d (TRS-80 3 & 4)	**$199**

How To Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

How To Get More Information

To get more information on Manx Aztec C and related products, call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

30 Day Guarantee

Any Manx Aztec C development system can be returned within 30 days for a refund if it fails to meet your needs. The only restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in resalable condition. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of competing systems. Call for information.

MANX

To order or for information call:

800-221-0440

UNIX is a registered TM of Bell Laboratories, Lattice TM Lattice Inc., C-tree TM Faircom, Inc., PHACT TM PHACT ASSOC., CI Optimizing C86 TM Computer Innovations, Inc., MACINTOSH, APPLE TM APPLE, INC., Pre-C, PLINK 86, TM PHOENIX, HALO TM Media Cybernetics, Inc., C-terp, PC-lint TM GIMPLE Software, Windscreen, SunScreen TM SunTec, PANEL TM Roundhill Computer Systems Ltd., WINDOWS FOR C TM Creative Solutions, XENIX, MS TM MICROSOFT INC., CP/M TM DRI, AMIGA, C64, C128 TM COMMODORE Int.

Circle **no. 108** on reader service card.

LETTERS

DOBB'S POST 2 CENTS 2

Not for Profit?

Dear *DDJ*,

Your most recent name change has brought into focus a matter which has been irritating me for some time: The copyright declaration stating that published programs are for "personal use only, not for profit." Your "Software Tools" are now said to be "for the professional programmer." Most professional programmers, in my experience, are in the business for profit (or at least to keep food on the table). Given the confusing (at least to me) state of litigation concerning software protection, even to the extent of reverse engineering now apparently illegal, I find myself wondering if I dare read any magazines for fear that I might unconsciously use a line of someone else's code for profit. I am particularly irritated when I note that the code in question has been copied largely from another source. K & R seems to be a rich source of freshly copyrighted material, for example.

A specific example which I found recently: An "author" took the code for GREP.C distributed by DEC, added *#defines* so that it would compile under another compiler, and copyrighted it as "not for profit," "cannot modify or distribute," and so on. Does this mean DEC can't use it anymore?

I would like to see further discussion of program protection and publication matters in *DDJ*. In the meantime, could you persuade some of your authors to state "portions may be used in applications providing suitable credit is given" or something to that effect?

Allen R. Balmer
6845 West Henrietta Rd.
Rush, NY 14543

DDJ has always been in favor of public-domain software; unfortunately, the line between public-domain and copyrighted code has become increasingly blurred. Even the laws seem a little confused at times. When a software author modifies a public-domain program and releases the modified version, is he or she entitled to copyright it? Does it depend on the amount and nature of the modifications made? We welcome your point of view.—ed.

Who is DDJ For?

Dear *DDJ*,

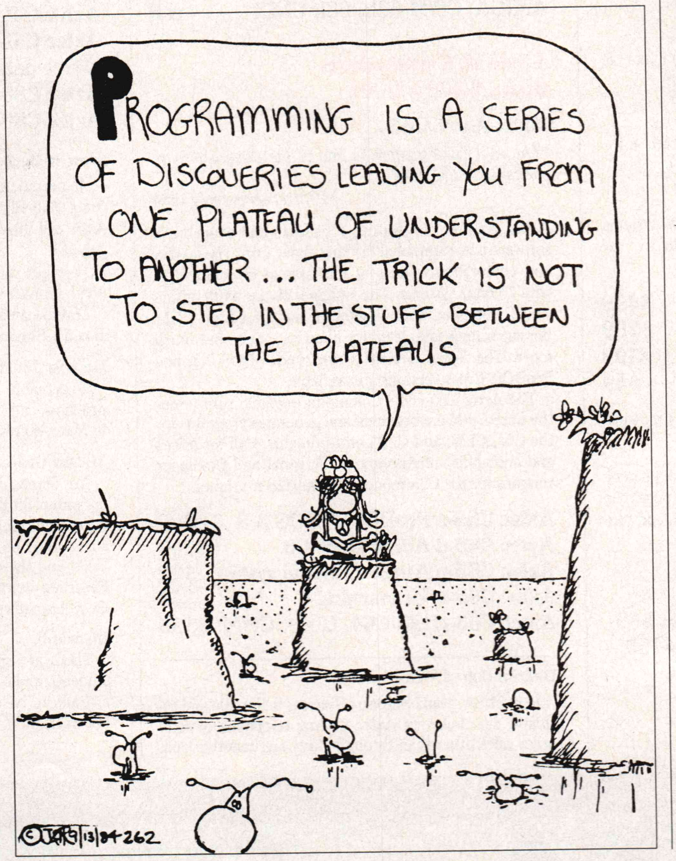
When I subscribed to *DDJ* three years ago, the magazine was described as "for users of small computer systems." A year later (March 1984), it became "for the experienced in microcomputing." Two months after that it became ". . . for advanced programmers." Now you are ". . .for the professional programmer." It seems that you are becoming "for" a smaller and smaller readership.

By some stretch of imagination I might still be considered eligible to be a subscriber, but your next redefinition will surely exclude me. But it probably won't matter, for by that time *DDJ* will likely be merely an index of what's available on your bulletin board.

Why not just publish the magazine and let the individual decide if the magazine is for him?

Dave Sullivan
207 Maclane St.
Palo Alto, CA 94306

As the market for computer magazines grows larger and more confusing, it becomes increasingly important for each magazine to carefully target its audience. DDJ's audience has always been a very devoted and well-defined group of people, and we'd like to stay focused on that group. This means that people who are looking for introductory articles on programming in BASIC or who need to learn how to use their new spreadsheet will have to look elsewhere. But it's also important not to get too carried away. We think Mr. Sullivan has a point. What do you think?—ed.

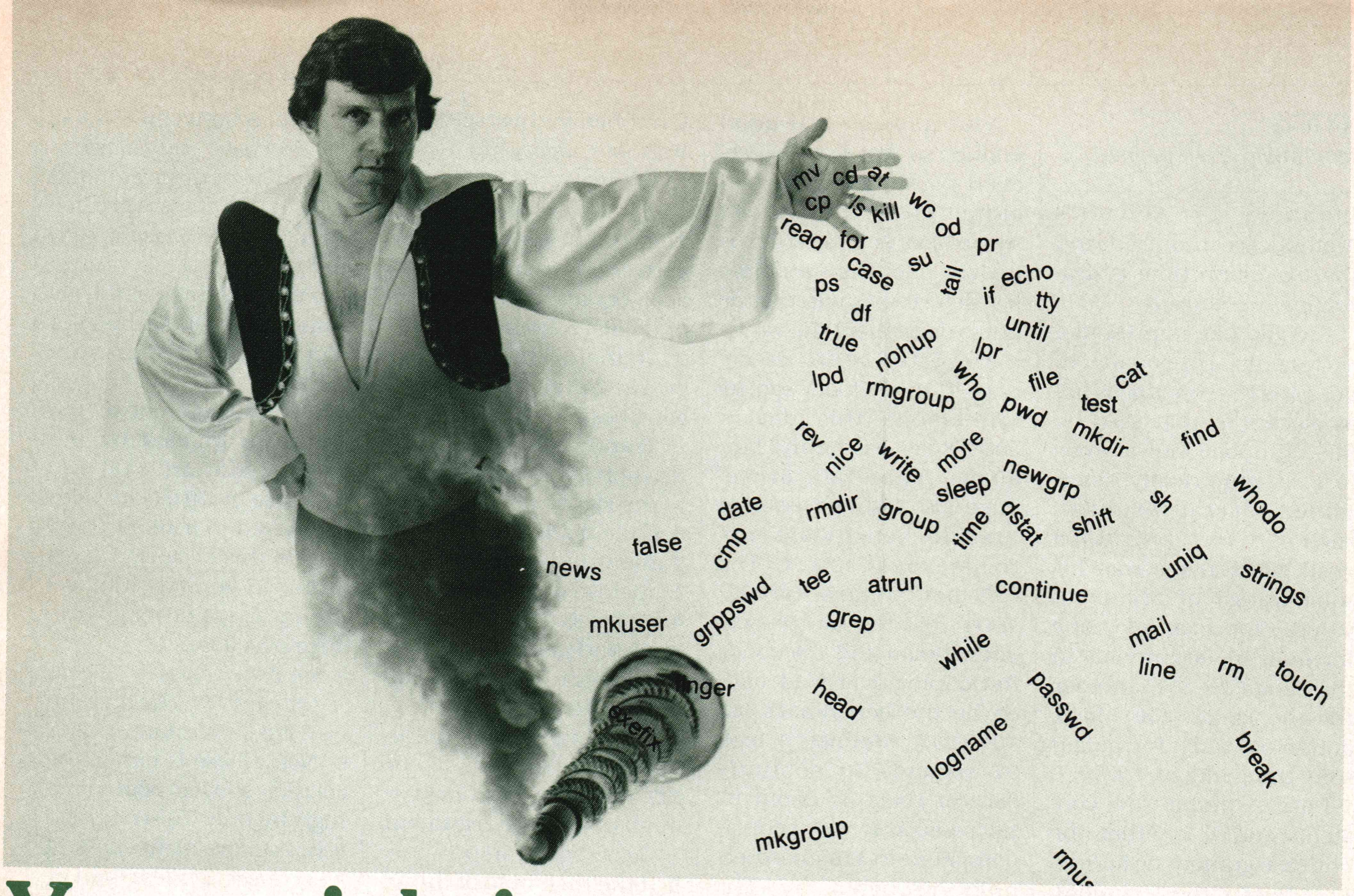


Right to Assemble

Dear *DDJ*,

Mr. Campbell's The Right to Assemble column (March 1986) on computing integer square roots presents a clever modification of Newton's Method that is considerably more efficient than previous implementations that have appeared in the pages of *DDJ* and other magazines.

The essence of Newton's Method is to guess the value of the root and then to compute successively better guesses, using a simple formula, until the desired accuracy has been achieved. Mr. Campbell uses an ingenious scheme to arrive at a good initial guess. As a result, very few



Your wish is our commands.

UNIX operating systems have a well deserved reputation for their powerful command set. Now PCUNIX gives you the same powerful commands. And a lot more.

Built with Wendin's Operating System Toolbox, PCUNIX is the only multitasking, multiuser operating system that can put the popular features of UNIX on your personal computer (IBM PC, AT, XT, or true compatible) for under $100.

If you're already familiar with the UNIX environment you know what that means. If you aren't, PCUNIX is your chance to discover what you've been missing at a price you can finally afford.

Designed specifically for systems programmers, PCUNIX has all the features that have made UNIX the preferred environment for systems development. Commands that let you perform complex operations with a few keystrokes. Plus syntax that lets you compile programs in the background, handle piping and I/O redirection, and build shell script files.

And that's not all.

PCUNIX links the popular Bourne shell to Wendin's unique kernel. As a result, you have access to more than eighty enhanced system services built into all our Personal Operating Systems.

Like our other Personal Operating Systems, PCUNIX also runs well-behaved MS-DOS programs, is fully compatible with the MS-DOS file system, and uses existing MS-DOS compilers, linkers and utilities.

In addition, it comes with complete source code that lets you see for yourself exactly how the system works.

So if you've ever wished you had the power and flexibility of UNIX on your PC, why not stop wishing and order PCUNIX today.

PCUNIX. From Wendin. Only $99.

Ask about our other products for the IBM PC and true compatibles.

PCVMS™
Multitasking, multiuser version of DEC's powerful VAX/VMS operating system. Runs most MS-DOS programs.

XTC®
The ultimate programmer's editor. Multitasking macro language plus multiple linkable windows and buffers.

OPERATING SYSTEM TOOLBOX™
Complete software construction set that lets you build your own multitasking, multiuser operating systems.

All products priced at $99 with source code included.

DEALER INQUIRIES WELCOME

Foreign orders inquire about shipping.
Domestic orders add $5.00/1st item,
$1.00 each additional item for shipping, handling, and insurance.
Washington residents add 7.8% sales tax.

MS is a trademark of Microsoft; PC-DOS is a trademark of IBM; UNIX is a trademark of AT&T; VAX/VMS is a registered trademark of Digital Equipment Corporation

WENDIN®
BOX 266
CHENEY, WA 99004

ORDER HOTLINE
(509) 235-8088
(MON.-FRI., 8-5 PACIFIC TIME)

VISA MasterCard

© Copyright 1986 Wendin, Inc.

The people who make quality software tools affordable.

Circle **no. 112** on reader service card.

Wendin and XTC are registered trademarks of Wendin, Inc. PCUNIX, PCVMS, Operating System Toolbox, and Personal Operating System are trademarks of Wendin, Inc.

LETTERS
(continued from page 8)

iterations of Newton's Method are required and the execution time is dramatically reduced.

I would like to present a different approach to guessing the root that is not as elegant as that used by Mr. Campbell but is even faster. I empirically determined several formulas, each of which provides a good guess at the root for some range of argument values (for example, one formula for arguments in the range 2–255, another for the range 256–4095, and so on). Each formula is something like $(A * X) + B$ where A and B are constants and X is either the entire argument or just the high byte of the argument.

The guesses are good enough so that a single iteration of Newton's Method either gives the correct square root or a value that is high by one. Then, a simple test determines whether or not to decrement the value.

The *CALC_ROOT* procedure (Listing One, page 60) implements this idea in 8088/8086 assembly language. The *TIME* procedure (also Listing One) was used to benchmark my routine on an IBM PC. The full benchmark (*TIME_ROOT* routine) ran 98 seconds. Deducting 7 seconds for looping overhead, added by the benchmark (*TIME_OVER* routine), it took 91 seconds to compute 983,040 roots or about 92 microseconds per root as compared to 183 microseconds reported by Mr. Campbell for his 16032 routine. My algorithm works as follows:

1. If the argument equals zero or one then the square root equals the argument and I just return its value.
2. If the argument is between 2 and 255 inclusive, set *Guess = Argument/16 + 3* and go to Step 7. The division by 16 is a 4-bit shift to the right.
3. If the high byte of the argument is between 1 and 15 inclusive, set *Guess = 4 ** (high byte of the argument) *+ 13* and go to Step 7. The multiplication is a shift left of two bits.
4. If the high byte of the argument is between 16 and 127 inclusive, set *Guess =* (high byte of the argument) *+ 50* and go to Step 7.
5. If the high byte of the argument is between 128 and 254 inclusive, set *Guess =* (high byte of the argument) *+ 40*. If this guess is greater than 255, set *Guess = 255* and go to Step 7.
6. If the high byte of the argument equals 255 then the root is 255 and I return that value.
7. Get the quotient of the argument divided by the guess. I use an 8-bit *DIV* which saves 70 clock cycles compared to a 16-bit *DIV*. I can do this because I have excluded (in Step 6 above) the one case that could produce overflow.
8. Set *New Guess = (Guess + Quotient)/2*. The division by 2 is a 1-bit rotate.
9. *New Guess* is either the correct square root or is high by one. To see which it is, I square *New Guess*. If *New Guess ^2* is less than or

We're Ready, Are You?

68000
68010
68020

WE ARE PROUD TO ANNOUNCE THE BIRTH OF THE NEWEST MEMBERS OF OUR 68000 FAMILY . . . YOUR 68020 TOOLS ARE HERE!

TOOL KIT
- 68000/10/20 Assembler Package:
  - Macro Cross/Native Assembler
  - Linker and Librarian
  - Cross Reference Facility
  - Symbol Formatter Utility
  - Object Module Translator
- Green Hills C 68000/10/20 Optimizing Compilers
- Symbolic Debuggers

FEATURES
- Written in C; fast, accurate, portable.
- Supports 68000 and 68010.
- 5,000 line test suite included.
- EXORmacs compatible.
- Produces full listings and maps.
- Outputs S-records and Tek-Hex formats.
- Runs native or cross. Extensive libraries.
- Supports OASYS compilers.
- Generates PROMable output and PIC.
- Full Floating Point support.

AVAILABILITY
VAX, microVAX, 8600, Sun, Pyramid, Masscomp, IBM/PC, OASYS Attached Processors for VAX and PC, others. Runs under VMS, Bsd 4.2, System V, MS/DOS, dozens more.
You name it . . .

We provide a "One-Stop Shopping" service for more than 100 products running on, and/or targeting to, the most popular 32-, 16- and 8-bit micros and operating systems.

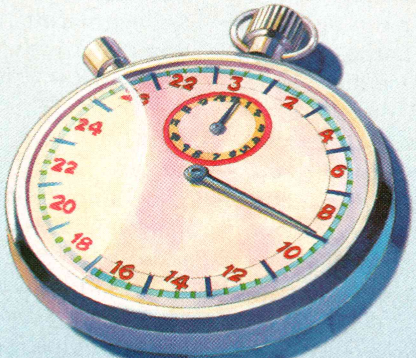
Oasys
A DIVISION OF XEL

We Specialize In:
Cross/Native Compilers C, Pascal, Fortran, Cobol, Basic, APL, PL/1, Prolog, Lisp, ADA — Assemblers/Linkers — Symbolic Debuggers — Simulators — Interpreters — Translators Converters — Profilers — QA Tools — Design Tools — Comm. Tools OS Kernels — Editors — Spreadsheets — Data Bases — VAX & PC Attached Processors and more
We Support:
680xx, 80x86, 320xx, 68xx, 80xx, dozens more

60 Aberdeen Avenue, Cambridge, MA 02138 **(617) 491-4180**

Circle **no. 254** on reader service card.

HOW TO TIME ALL 53,110 PROCEDURE CALLS IN 9 SECONDS.



The mystery of all time.

It's the biggest problem facing every designer of embedded microprocessor-based software: How can you tell exactly where your code is spending its time during execution in the target environment?

Most current systems for analyzing software can only give you a statistical answer at best. Or worse, an answer falsified by the fact that you had to modify your software in order to view it.

In performance analysis, there's no time like the real time.

NWIS has the answer. The SoftAnalyst™ provides the first real-time performance analysis of your software. You measure segments of interest within your code, using the same symbols defined in your source code. For example, you can specify procedure entry and exit points, line numbers, or code ranges. The SoftAnalyst shows you how frequently each segment executed, and how much total time it consumed. It also shows true minimum and maximum execution times so you can know if your real-time code is ever out of spec.

And it's non-intrusive, so you don't have to modify your code to measure it.

Finally, it's non-statistical; automatically tracking every single entry to and exit from your specified areas of code. It's language-and-compiler-independent, working equally well on C, PASCAL, FORTRAN, or assembly; on code compiled on computers ranging from VAXes to PCs.

Software analysis has never been smarter.

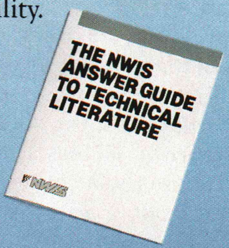
The SoftAnalyst does much more than just performance analysis.

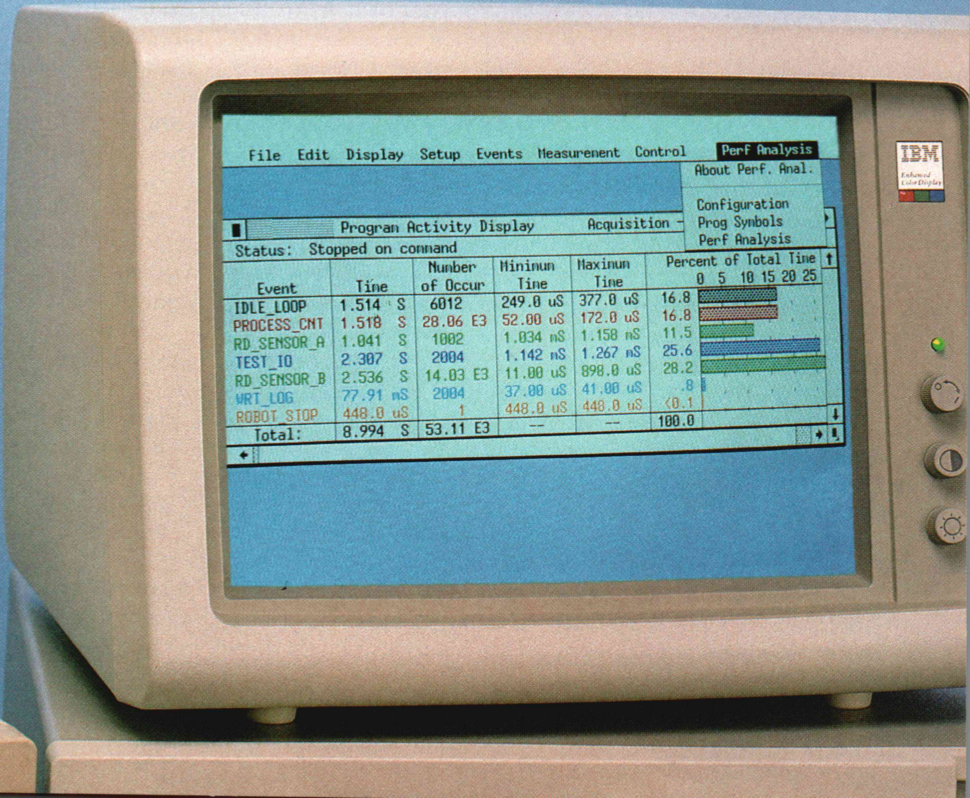
It also gives you CodeMap,™ which monitors your address space, and tells you whether or not your test cases exercised all of your code.

And SymTrace™ provides high-level symbolic monitoring of your program and data flow.

We'll show you how:

Call us at **800-547-4445** to schedule a SoftAnalyst demonstration at your facility. Or send for our free publication "The NWIS Answer Guide to Technical Literature."





NWIS

NORTHWEST INSTRUMENT SYSTEMS, INC.
P.O. Box 1309 · Beaverton, OR 97075
800-547-4445

For Literature circle no. 232 on reader service card.
For Demonstration circle no. 235 on reader service card.

(continued from page 10)

equal to the argument, then the square root equals *New Guess* or else it equals *New Guess* − 1.

This method is an example of perspirational rather than inspirational programming. I wrote a small BASIC program (Listing Two, page 62) designed to test out different formulas. After some tedious hours trying various possibilities I found a set of ranges and formulas that worked.

My BASIC program is quite fast because it only tests argument values that are one less than a perfect square (for example, numbers of the form N^2 − 1). These tend to be the worst case situations because the exact root of such numbers is very near to N while the integer root is $N - 1$.

The *TEST* procedure (Listing One) was used to verify the accuracy of the root for all 65536 argument values. In this test, I checked that the square of the root is less than or equal to the argument and that *(root + 1)*^2 is greater than the argument.

Robert Pirko
211 West 56th St.
Apt. 36L
New York, NY 10019

Dear *DDJ*,
I noticed The Right to Assemble column discussing integer square roots as implemented on the 68000 and the 32000 in the March 1986 issue. I wish to further illustrate the advantage of using the higher-level instructions of the 32000 with a listing of my routine to perform a floating-point square root in software. It uses a format similar to IEEE double precision, except that I put the exponent on the right to reduce the number of operations required. My 16032 with a 7.16-MHz clock takes 190 microseconds to do the double-precision square root (including variable passing overhead). This is 1,113 times faster than BASCOM does it on my Z80 system (3.68 MHz, 1 wait state). Because I don't have access to a 68000 computer, I don't know how it would fare in this task.

In Listing Three, page 62, you will notice there is no looping and there are no calls to the floating-point multiply or divide routines. The algorithm used is called Newton's Method and iteratively executes the following equation:

Y = Y/2 + (X/2)/Y

The equation is based upon the use of a derivative to extrapolate and calculate a more precise guess. The binary exponent is a great aid in choosing the first guess. We know to start with that the mantissa of the answer will be between 1 and 2; furthermore, the even-or-odd condition of the original exponent tells us if the mantissa of the answer will be above or below the square root of 2. With this guidance, we make the first guess either 1.189 or 1.68. This is enough of a start so that four iterations is always enough to obtain the required 53 bits of precision. Only in rare cases would three iterations be enough, so we always do four iterations without any testing. Because the precision of the result more than doubles with each iteration, the first three iterations can be performed with less precision, and the compactness of the 32000 instructions makes it easy to take advantage of this fact. Notice in the listing, starting at label *SQR2*, how easy it is to do the first three iterations with 32-bit precision on the 2000. The *DEID* R3,R4 instruction takes a 64-bit value in R4R5 and divides it by the 32-bit value in R3. It puts the quotient in R5 and the remainder in R4. Four machine-code instructions to evaluate the whole equation is what I would call efficient.

Neil R. Koozer
Kellogg Star Rt.
Box 125
Oakland, OR 97462

Help

Dear *DDJ*,
We are a nonprofit organization founded to help the handicapped and disabled acquire job skills and jobs with the assistance of PCs. Our staff helps coordinate the efforts of rehabilitation agencies and educational institutions. Whenever possible, we supply the needed PC equipment at no cost and train people to use it.

To disseminate information and reach and encourage the handicapped and disabled, we publish a bimonthly newsletter *Personal Computer Opportunities for the Handicapped*. If you wish to subscribe, please send your name and address. We ask that you make a contribution of a minimum of $10. This contribution is fully tax deductible. If you are handicapped or prefer not to make a contribution, drop us a note and we will give you a subscription at no charge.

We deeply appreciate any assistance you can give us.

James R. Nichols,
Personal Opportunities
for the Handicapped
P.O. Box 374
Spicer, MN 56288

Corrections

Dear *DDJ*,
Thank you for including our software in the March 1986 issue of *DDJ*. There is a slight error in that paragraph which depreciates the value of our hardware. It should read ". . . LAB 40 is a structured parallel port that takes 16 memory or I/O locations . . ."

The article has it reading 16K (the K should be omitted).

Scott Vanderlip
75 Southgate Ave.
Suite 6
Daly City, CA 94015

Dear *DDJ*,
By this time you have undoubtedly caught the errors in The Right to Assemble column in March 1986.

The worst error is the definition of the (integer) square root of an integer, at the top of column 2. By your definition, the square root of 17 (or of any prime number) would be 1.

The others, at the top of column 3, lead you to say in one sentence that "4 is not the correct root" and in the next that "the integer square root of 24 is 4." In the same sentence, you have "24 divided by 4 equals 5."

Dorothy Wolfe
245 Hathaway Ln.
Synnewood, PA 19096

Dear *DDJ*,
Thank you for your excerpt on our company in your January (1986) Of Interest column.

We would like to point out that you mentioned our company name as ATC International in your column when actually it should have been ACS International.

Janet M. Heidenreich
Advanced Computer
Solutions International
2105 Luna Rd.
Suite 330
Carrollton, TX 75006

DDJ

(Listings begin on page 60.)

Turbo Pascal and the Turbo Pascal family give you a perfectly integrated programming environment and unbeatable speed, power, and price

Turbo Pascal® is *faster* than any other Pascal compiler, and at only $69.95, a distinctly better deal. But it offers much more than speed, power, and price.

There's also the complete Pascal family of products that's grown from 1 to 9 products in just 3 years.

- Turbo Pascal 3.0
- Turbo Pascal with the 8087 support
- Turbo Pascal with Binary Coded Decimal, (BCD)
- Turbo Pascal with 8087 and BCD
- Turbo Database Toolbox™
- Turbo Graphix Toolbox™
- Turbo Tutor®
- Turbo Editor Toolbox
- Turbo GameWorks

Turbo Pascal is backed by a complete range of "toolboxes" that give you most of the programming tools you'll ever need.

The Turbo Pascal family is never static, but is continuously expanding, with new products like Turbo Editor Toolbox™ and Turbo Gameworks.™

The secret of software success is not merely low price, but top quality, allied with complete documentation, like our 400-page reference manual.

All of which are some of the reasons why Turbo Pascal is clearly the leader, and the recipient of awards like PC Week's **"Product of the Year"** and PC Magazine's **"Award for Technical Excellence."** And some of the reasons why Turbo Pascal has now become a *de facto* worldwide standard with more than half a million users.

Turbo Pascal has grown from a single product 3 years ago to a family of 9 today.

Success breeds success, so the Turbo Pascal family has flourished. Your choices now include:

☐ **Turbo Pascal 3.0** combines the fastest Pascal compiler with an integrated development environment.

☐ **Turbo Pascal with 8087** math co-processor support for heavy duty number-crunching, and/or **Binary Coded Decimals** to eliminate rounding-off errors for business applications.

☐ **Turbo Database Toolbox** is a perfect complement to Turbo Pascal. It includes a complete library of Pascal procedures that allows you to search and sort data, and build powerful database applications.

☐ **Turbo Graphix Toolbox** includes a library of graphics routines for Turbo Pascal programs. Lets even beginning programmers create high-resolution graphics with an IBM,® Hercules,™ or compatible graphics adapter. Does complex business graphics, easy windowing, and stores screen images to memory.

☐ **Turbo Tutor** teaches you step by step how to use Turbo Pascal, with commented source code for all program examples on diskette.

Save $109.70 when you choose the Turbo Jumbo Pack. 6 different Turbo Pascal products for only $245.00!

For only $245.00, you get Turbo Pascal 3.0 *and* Turbo Editor Toolbox *and* Turbo Tutor *and* Turbo Graphix Toolbox *and* Turbo GameWorks *and* Turbo Database Toolbox!

All 6 for only $245.00, which saves you $109.70. This limited offer is good through September 1, 1986, so act now.

NEW! Amazing value! Turbo Editor Toolbox includes MicroStar,™ a full-blown editor that also does windows! **Turbo Editor Toolbox** not only gives you ready-to-compile source code and a 200-page manual that tells you how to integrate the editor procedures and functions into your programs, but also includes MicroStar, a complete editor with full windowing capabilities. (You could pay $100.00 or more for a program like MicroStar, but you get it free as part of our Turbo Editor Toolbox.) You can also use Turbo Editor (which of course integrates with Turbo Lightning™) to build your own word processor!

NEW! Turbo GameWorks gives you the games you can write, rewrite, bend and amend! **Turbo GameWorks** reveals the secrets of game design and the strategies. You're given source code, a 200-page manual, and the insight needed to write and customize your own irresistible games.

Turbo GameWorks also includes ready-to-play Chess, Bridge, and Go-Moku—an ancient Japanese game that can divert you from reality for hours on end.

"Language deal of the century . . . Turbo Pascal

Jeff Duntemann, PC Magazine

Turbo Pascal has got to be the best value in languages on the market today

Jerry Pournelle, BYTE Magazine

This compiler, produced by Borland International, is one of the best programming tools presently available for the PC"

Michael Covington, PC Tech Journal

YES! I want the best

To order by phone, or for a dealer nearest you,

call (800) 255-8008

in CA call (800) 742-1133

Copies	Product	Price	Totals
___	Turbo Pascal 3.0	$69.95	$ ______
___	Turbo Pascal w/8087††	$109.90	$ ______
___	Turbo Pascal w/BCD††	$109.90	$ ______
___	Turbo Pascal w/8087, BCD††	$124.95	$ ______
___	Turbo Database Toolbox	$54.95	$ ______
___	Turbo Graphix Toolbox†	$54.95	$ ______
___	Turbo Tutor	$34.95	$ ______
___	Turbo Editor Toolbox†	$69.95	$ ______
___	Turbo GameWorks†	$69.95	$ ______
___	Turbo Jumbo Pack†	*$245.00	$ ______
	Outside USA add $10 per copy CA and MA res. add sales tax		$ ______
	Amount enclosed		$ ______

Prices include shipping to all US cities.

Carefully describe your computer system:

Mine is: ___ 8-bit ___ 16-bit

I use: ___ PC-DOS ___ MS-DOS ___ CP/M-80 ___ CP/M-86

My computer's name and model is: ______________________

The disk size I use is: ☐ 3½" ☐ 5¼" ☐ 8"

Payment: VISA MC Bank Draft Check

Credit card expiration date ___/___

Card # ______________________

NOT COPY PROTECTED TF11

****60-DAY MONEY-BACK GUARANTEE**

Name: ______________________

Shipping Address: ______________________

City: ______________________

State: ____________ *Zip:* ____________

Telephone: ______________________

CODs and purchase orders WILL NOT be accepted by Borland. Outside USA make payment by credit card or International Postal Money Order.

**Limited Time Offer until September 1, 1986.*

***YES, if within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.*

Minimum System Requirements:

Turbo GameWorks, Turbo Graphix Toolbox, & Turbo Editor Toolbox—192K. All other products, 128K.

†IBM PC, PCjr, AT, XT, and true compatibles.

††16-bit only.

BORLAND INTERNATIONAL

Vive la différence

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

Borland products include Turbo Pascal; Turbo Prolog; Turbo Database Toolbox; Turbo Lightning; Turbo Graphix Toolbox; Turbo Tutor; Turbo GameWorks; Turbo Editor Toolbox; Word Wizard; Reflex, The Analyst; SideKick; Sidekick, The Macintosh Office Manager; Traveling SideKick; and SuperKey—all of which are trademarks or registered trademarks of Borland International, Inc. or Borland/Analytica, Inc.

Turbo Pascal and Turbo Tutor are registered trademarks, and Turbo GameWorks, Turbo Editor Toolbox, Turbo Database Toolbox, Turbo Graphix Toolbox, Turbo Lightning, and MicroStar are trademarks of Borland International. IBM is a registered trademark of International Business Machines Corp. Hercules is a trademark of Hercules Computer Tech.
Copyright 1986 Borland International. BI-1036C

Circle **no. 161** on reader service card.

DDJ ON LINE

For those of you who missed it the first time, here is how to log on to the DDJ Forum.

1. Get yourself a CompuServe account. (Call CompuServe at (800) 848-8199 for information.)
2. Log on to CompuServe and type go ddj at the prompt. This will bring you into the Display Area, the Read Only (noninteractive) module of the SIG. Here you'll find information about the DDJ Forum.
3. Select the last menu choice from the Display Area Menu. This will bring you into the Forum where the message boards, Data Libraries, and real-time conferencing features are. That's also where you'll find me and columnists such as Mike Ham, Namir Shammas, Ray Duncan, and Allen Holub.
4. Type I at the Top Menu. This will give you a list of commands and a thumbnail sketch of the Forum's structure and features. You should capture this list and print it out. It's handy to use as a map until you acquaint yourself with the territory.
5. Read the bulletins and help files. Type B at the Top Menu to read the bulletins. These provide more information about various aspects of the Forum. I also recommend that you go to DL0 (type DL0 at the Top Menu) and browse the help files there. To do this, type BRO at the DL menu. When it asks you for KEYWORDS, type HELP. The best help file for a quick start is EZSIG HELP.

*That should get you started folks. Of course the only real way to learn how the system works is to work on the system <grin>. Try things out, make a few mistakes, get the feel of it. And if all else fails, ask that *SYSOP person—she'll be glad to help.*

*—Levi Thomas (*SYSOP)*

The following is an on-line exchange that took place in the Forum's House of ALGOL (the on-line version of Namir Shammas' Structured Programming columns).

Modula-2

11-Mar-86
Sb: Modula-2 Tools
Fm: Bill 73047,2624
To: All
I am just getting started in Modula-2 although I have programmed in just about everything else. I am writing a simple multitasking dispatcher for a fun project I'm developing. When I get it to where I like it, I'll let you know if you're interested. I suspect that I'm reinventing the wheel, but I haven't been able to find anything like it in the public arena.

12-Mar-86
Sb: Modula-2 Tools
Fm: Bob 76703,532
To: Bill
I assume you are aware of MODUS, the Modula-2 User Group? If not, I'll gladly provide you with an address. Check out recent issues of ACM *SIGPlan Notices*. (It should be available at most libraries.)

The June 1985 issue contains two Modula-2 articles. One of them, entitled "Two Approaches to Implementing Generic Data Structures in Modula-2," is by Weiner and Sincovec, the gentlemen with a reasonable Modula-2 book to their credit.

The December 1985 issue contains three articles relating to Modula-2, including one from a user of both Modula-2 and Ada.

The articles you will be most interested in are entitled "Modula-2 Process Facilities" and "Modula-2 and the Monitor Concept." These are by D. A. Swery and are in the November 1984 issue. Both articles are reasonably well written and both contain source code. The first article extend's Dr. Wirth's binary semaphores to counting semaphores, and the second is a Modula-2 implementation of Hoare's monitor concept. That both semaphores and monitors can easily be implemented in Modula-2 (without resorting to assembly) is an excellent example of the power of Modula-2's coroutines.

The October 1984 issue has an article about implementing semaphores under Unix without kernel changes. Some code (in C) is included as examples. This article is of interest because it makes minimal assumptions about the underlying OS.

The January 1986 issue contains an article entitled "Detection of Deadlocks in Multiprocess Systems," which should be of interest to people designing tasking systems.

13-Mar-86
Sb: Modula-2 Tools
Fm: Bill
To: Bob
Thanks much for the info. I'll check in the company library for the *SIGPlan Notices*. Could you please give me the address for MODUS? I'm afraid I'm a Pascal bigot and ignorant of the Modula-2 user community. I'm rapidly converting! Would there be a more definite way of asking for the SIGPlan info? Sometimes the library isn't too sure of the really technical stuff. Thanks again!

14-Mar-86
Sb: Modula-2 Tools
Fm: Bob
To: Bill
The address of MODUS is
MODUS
c/o George Symons
P.O. Box 51778
Palo Alto, CA 94303
(415) 322-0547
Dues are $20/academic year. The basic benefit is four newsletters/year. The membership form asks for name, company (if appropriate), address, phone, electronic address, and which implementations of Modula-2 you use. In addition there are three options:

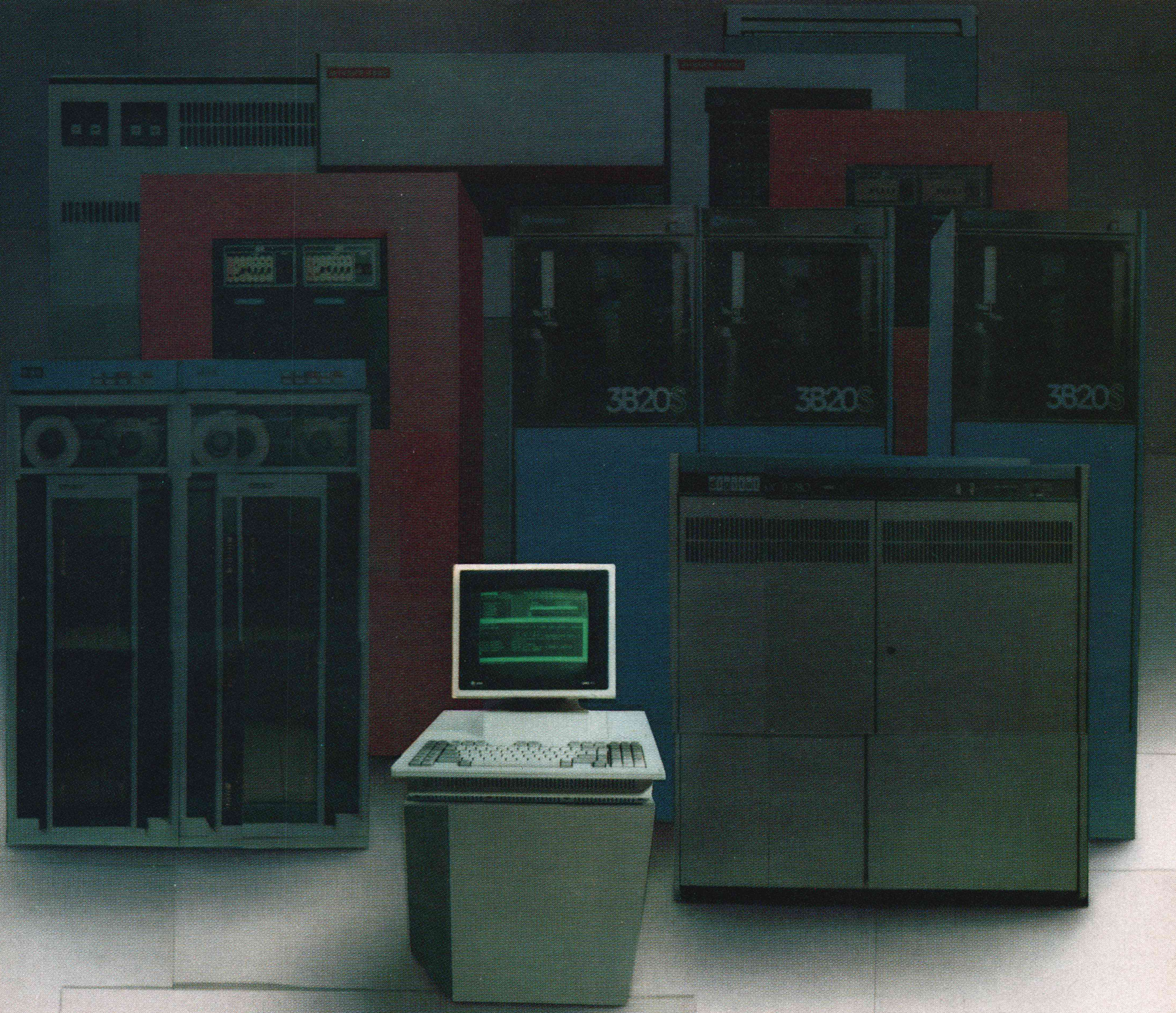
1. Don't print my phone number on MODUS rosters.
2. Print only my name and country in any rosters.
3. Don't release my name on any mailing lists.

When you join, you'll be sent all back issues for the current year.

Your librarian should be able to look up the *SIGPlan Notices* issues using the dates given. *SIGPlan Notices* is the actual name of the magazine. It is a monthly publication of ACM's (Association for Computing Machinery) Special Interest Group for Programming Languages.

If your company library doesn't have *SIGPlan Notices*, many public libraries have it. It really is quite common.

20-Mar-86
Sb: Logitech's Modula-2
Fm: Bill
To: Bob
I've just acquired Logitech's Modula-2. I had the ITC (still



UNIX™ SYSTEM POWER FOR PEOPLE WITH BIGGER THINGS IN MIND.

You know what **UNIX**™ System V can do.

But now you don't need a mini to do it. The AT&T **UNIX** PC puts room-size computing power right on a desktop.

Its Motorola 68010 chip, 10 MHz clock speed and up to 4MB RAM—with virtual memory support and internal hard disk options from 10 to 67MB—give you 75% of the power of a VAX* 11/780.

For only 7% of the cost.

Development tools? The AT&T **UNIX** PC puts you in a **UNIX** System V environment complete with system utilities, the shell, C compiler and 68010 assembler. As for languages, you get the full range: C, Cobol, Fortran, Pascal, BASIC and the LPI** high-performance suite. Not to mention C-ISAM†, INFORMIX† and sort/merge for database development.

All, with the convenience of built-in text editors, debuggers and graphics tools, including the GSS Virtual Device Interface.

Up- and downloading your work from minis or mainframes is easy. Thanks to the standard internal 300/1200 bps modem, RS-232 port, VT 100* terminal emulation software and optional 3270 terminal emulation. You also get two jacks for phone lines and built-in communications software.

All of which make the AT&T **UNIX** PC ideal for ongoing voice/data communications and remote access to shared corporate databases.

ONE OF THE COMPUTERS WITH THE FUTURE BUILT IN.

Even with all its available power and storage options, the AT&T **UNIX** PC still has room to grow. With three expansion slots and the ability to connect up to seven serial devices.

Because when you have big ideas, accommodating them shouldn't be a big deal.

To find out about the AT&T **UNIX** PC and our **SPECIAL LIMITED TIME OFFER** call your AT&T Account Executive, authorized AT&T supplier or 1 800 247-1212.

AT&T
The right choice.

*VAX and VT100 are trademarks of Digital Equipment Corporation. **LPI is a trademark of Language Processors, Inc.
†C-ISAM and INFORMIX are trademarks of Relational Database Systems, Inc. © 1986 AT&T Information Systems.

DDJ ON LINE
(continued from page 14)

have it, actually) and found it much too buggy. So far the only problem I've found with Logitech is that I didn't go for the "sources" right off. But . . . that's for when the checkbook gets back to a plus balance I guess. On a similar subject, what are all the Modula-2 folks reading out there? I just ordered the Wirth third edition from the publisher, but I suspect that other good Modula-2 books are available. I have the "Shipbuilder's" book but found it to be much too low a level. As a lifetime systems programmer, I need meat to keep me happy—none of this "cute" stuff. (Do "real systems programmers" read "cute"?) I noticed that McGraw-Hill (in the CServe electronic mail) had Olglvie(sp?)'s book for sale. Is it any good? Any other suggestions?

19-Mar-86
Sb: Turbo Multitasking
Fm: Jean 76606,671
To: Sysop—All
Hi! I bought my first *DDJ* last week (never too late. . .). I found the paper about multitasking with Turbo-Pascal very interesting. I looked for an IBM XT version in your DLs. Can I expect to see it one of these days? I'm a new IBM user (I'm more used to Apple-Pascal) and I didn't find all addresses to drive the modem. Also, does someone here know if Logitech's Modula-2/86 Version 2.0 compiler (the one announced for $89) supports all Modula-2 features (especially coprocessing)?

19-Mar-86
Sb: Logitech's Modula-2
Fm: Steve 70003,1326
To: Jean
Yes, it is all there. I have been working with Logitech's Modula-2 for about a month now. It is for real, a production compiler. I have it running on my DEC Rainbow (the editor is only for the IBM PC). I have already written a simple terminal program in Modula-2. I have no problem getting to INTs and interrupt vectors.

20-Mar-86
Sb: Logitech's Modula-2
Fm: Bob
To: Jean
Yes! Logitech's Modula-2/86 does support the entire language, including coroutines, *TRANSFER*, and *IOTRANSFER*. (*IOTRANSFER* implies being able to write interrupt handlers in Modula-2, which can in fact be done using Logitech's compiler.)

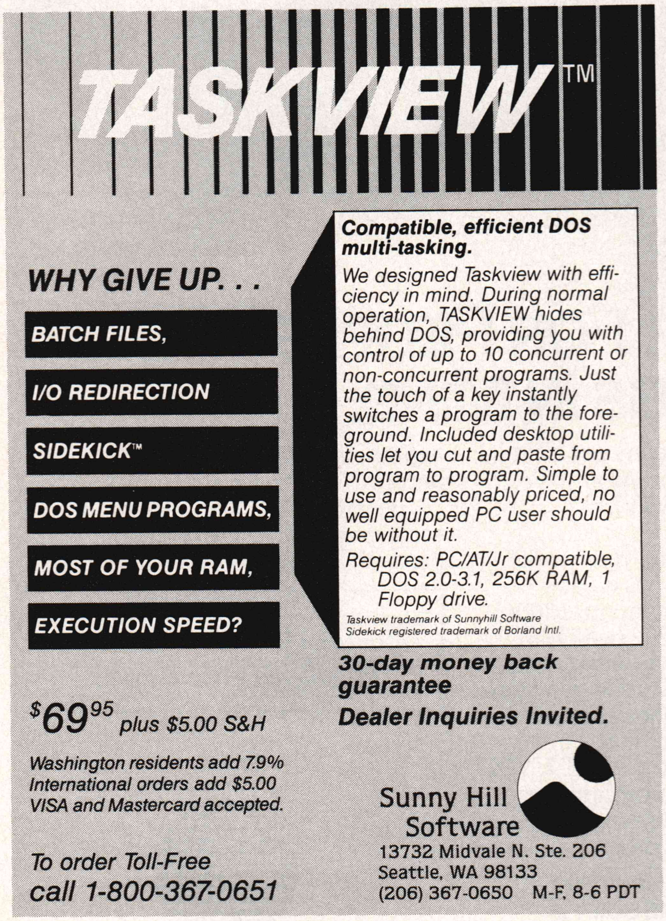
Logitech's Modula-2/86 is probably the best Modula-2 for the IBM PC, clones, and the 8086 in general.

21-Mar-86
Sb: Logitech's Modula-2
Fm: Bill
To: Jean
I have the debuggers—Post and Runtime. Both are unreal and well worth every cent. I, too, forewent the "sources" and will order them as soon as the checkbook recovers from all the other things I have gotten in the last few weeks. Although you don't "need" them to run the compiler, they would be nice as it is through the sources that you can customize the compiler/editor/linker options. It would also be nice to see the internals of some of the library modules. (I am, alas, a "real" systems programmer and not seeing the real code deprives me of my satisfaction.) I hope that this forum on Modula-2 gets going as I am just learning this language (it's probably number 20 or so) and it's the best so far. Now, if somebody will just point me to an Ada compiler environment for the PC, I'll move on (hee, hee). . . . You must understand, I dearly love my "toys" (got it from my father) and technical stuff gets me high.

20-Mar-86
Sb: Logitech's Modula-2
Fm: Jean
To: Steve
Thanks to Steve, Bob, and Bill for your answers! Great feedback in this SIG! So, think I'm gonna order that Modula-2. An IBM SW user suggested I buy the Utility package (for Postmortem debugger) and Sources features. Sources are expensive . . . I don't know. . . so I'm happy to find help. I'll make my first steps in that great language, the normal step after Pascal. So, thanks again! Jean (ps: Are you all "real system programmers"?—I am an amateur programmer!)

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 1**.



Circle **no. 172** on reader service card.

QUIT DOING GRUNT WORK.

Let Greenleaf do it for you and set you free.

C Program developers, stop slaving!

Greenleaf libraries have the functions you need — already perfected and in use by winning program developers in major corporations such as IBM, EDS and GM.

Between our Greenleaf Functions and Greenleaf Comm Library, we have over 340 functions on the shelf. Each one can save you time and effort. Money, too.

Many C programmers have told us that, even if they only use one or two functions, our products easily pay for themselves:

The Greenleaf Functions

The most complete and mature C language function library for the IBM PC, XT, AT and close compatibles. Our version 3.0 includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, new disk status and Ctr1-Break control functions plus many more!

The Greenleaf Comm Library

Our 2.0 version is the hottest communications facility of its kind. Over 120 all new functions — ring buffered, interrupt-driven asynchronous communications.

Call Toll Free

1-800-523-9830

In Texas and Alaska, call

214-446-8641

GREENLEAF

Software

Greenleaf Software, Inc.
1411 LeMay Drive Suite 101
Carrollton, TX 75007

If you need more than 2 ports, only Greenleaf gives you the total solution — boards, software, and complete instructions that enable you to build a 16-port communication system.

And no matter how many ports you have, it's virtually impossible to lose information with multiple file transfers. XMODEM, XON/XOFF and Hayes modem controls are featured.

We support all popular C compilers for MS DOS: Lattice, Microsoft, Computer Innovations, Wizard, Aztec, DeSmet and Mark Williams.

Order today!

Order a Greenleaf C library now. See your dealer or call 1-800-523-9830. Specify compiler when ordering. Add $8 for UPS second day air, or $5 for ground. Texas residents, add sales tax. Mastercard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf Comm Library v2.0	$185
Greenleaf Functions v3.0	$185
Digiboard Comm/4-II	$315
Digiboard Comm/8-II	$515

We also sell compilers, books and combination packages.

Circle **no. 97** on reader service card.

Trees and More on Microsoft and Lattice Compilers

My main topics this month are binary trees and C compilers. I'll look at a couple of fancy tree-printing routines and at a nonrecursive tree-traversal routine. Before leaping into trees, though:

A New Version of the Shell

DDJ is now shipping a new version of the shell that originally appeared in this column. In addition to fixing all the bugs I know about, I've included several significant enhancements in the new version. Pipes are now supported (you can even put the pipe temporary files on a RAM disk if you want). The *alias* and *history* expansion routines have also been improved considerably. You can now say:

```
a foo echo foo
a bar echo bar
a foobar 'foo;bar'
```

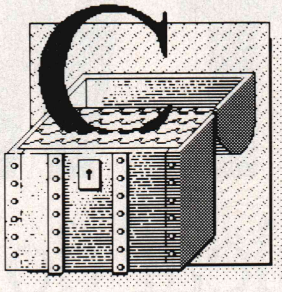
as well as:

```
!! >bar; !pat
```

DOS-compatible prompt support has been added (*$t*, *$d*, *$e*, and so on). You can change the escape character from backslash to another character so you can use backslash as a path separator. *Exit* takes a value so that a batch file that was called as a subroutine from another batch file can return a value to the calling process. *$status* is supported (it works a bit like *errorlevel* does).

by Allen Holub

Most important, all the C shell control flow commands (except *goto*) are now supported. In particular, there is an *IF . . . THEN . . . ELSE* mechanism, *WHILE* and *FOREACH* loops, and a C-like *SWITCH* statement. Complicated expression analysis is supported in these statements, using several operators (( ), +, −, *, /, %, <=, >=, <, >,



!=, ==, !, &&, and | |). You can create and modify variables (with an @ command). In short, you can now actually write real shell scripts.

Upgrades from Version 1 are available from *DDJ* for $6.

Printing a Tree

Ever since I was an undergraduate, I've wanted to write a routine that printed binary trees graphically—that drew a picture of the tree, showing with dashes and arrows where all the pointers went and which nodes were where. So this month, I finally sat down and did it. In fact I did it twice—once for an in-order traversal and once for a preorder traversal. Output from the tree-printing routines is shown in Figures 1 and 2, page 19.

The tree-printing routines are in Listing One, page 68. The *sinorder( )* function (lines 70−94) is included to show how the basic algorithm works. The static variable *depth* keeps track of the depth of the current node in the tree. The root is at depth 0, its children are at depth 1, and so on. The subroutine does an in-order traversal in the normal way. Instead of just printing each node, however, it prints *depth* tabs and then prints the node. This way, the farther down a node is in the tree, the farther to the right it will appear on the page.

A sample output is shown in Figure 3, page 19. There are two problems here. First, because there are no connecting lines, it's a little hard to see the internal connections in the tree. Second, a mirror image of the tree has been printed. Because a normal in-order traversal looks like:

```
traverse( root )
{
    traverse (left)
    print the root
    traverse (right)
}
```

the leftmost node of the left subtree will be printed first. A glance at Table 3 shows that the output is backward—the leftmost node ends up on the far right.

Both these problems are corrected in the subroutine *inorder* (lines 120−156). Fixing the mirror-image problem is easy. You just change the traversal algorithm to:

```
traverse( root )
{
    traverse (right)
    print the root
    traverse (left)
}
```

Getting the lines is a little more difficult. The problem is the vertical lines (printed with | characters). A bit map is kept in which each bit corresponds to a particular depth in the tree. If a bit is set, then a | is printed when you arrive at the equivalent depth in the output. So, all you need to do is set and clear these bits at the appropriate time. The relationships between the bit map and the final picture are illustrated in Figure 4, page 19.

If you set and clear the bits in the simplest possible way (that is, set the bit on line 131 of Listing One and clear it on line 153), a picture such as:

```
    |       +---g
    +---f---+
    |       +---e
d---+
    |       +---c
    +---b---+
    |       +---a
```

is created. You can avoid the topmost line by adding horizontal lines as you ascend rather than descend the tree. That is, the horizontal line for node *f* won't be added until after you've processed node *g*. This is done in the code on line 147 by setting the bit for the current level after a node has been printed. Setting the next level if there is no right child (line 135) avoids problems such as:

```
    +---g---+
    |           +---f
    |       +---e---+
d---+
```

and

```
f---+
            +---e
    +---d---+
```

where a line is omitted.

There are two situations in which a bottom extraneous line is created:

```
            +---i
    +---h---+
g---+
|   |       |   +---f
    |       +---e---+
    +---d---+
    |       +---b
```

To get rid of the extraneous line to the left of the *b*, you need to know whether the current node is a left or right child. You can then clear the appropriate bit when you process a left child. In the above example, if node *d* knows that it's a left child, it can clear the bit at its own depth before descending. This way the extra line isn't printed to the left of the *b*. The variable *amleft* indicates whether the current node is a left or right descendant. It should be set to 0 the first time *inorder( )* is called.

The second extra line (to the left of the *f*) is actually left over from processing node *i*. Because there was no left child, the test I just described wasn't performed. This situation is addressed by the code on lines 149—152. If there's no left descendant for the current node, you'll always clear the bit at *depth+1*. In the above example, because node *h* has no left descendant, it will clear the extra bit before ascending. The problem of no right child is addressed on lines 132—135.

The *preorder( )* routine (lines 172—203) uses more or less the same process. A new problem pops up here. You need to print an occasional blank line so that the output doesn't

```
                    +---O
            +---N---+
            |       +---M
    +---L---+
    |       |       +---K
    |       +---J---+
    |               +---I
H---+
    |               +---G
    |       +---F---+
    |       |       +---E
    +---D---+
            |       +---C
            +---B---+
                    +---A

H
+-------D
|       +-------B
|       |       +-------A
|       |       +-------C
|       +-------F
|               +-------E
|               +-------G
|
+-------L
        +-------J
        |       +-------I
        |       +-------K
        |
        +-------N
                +-------M
                +-------O
```

Figure 1: *Printing a balanced tree*

```
                    +---N
            +---M---+
    +---L---+
    |       |       +---K
    |       +---J---+
    |               +---I
H---+
    |               +---G
    |       +---F---+
    |       |       |       +---E
    |       |       +---D---+
    |       |               +---C
    +---B---+
            +---A

H
+-------B
|       +-------A
|       +-------F
|               +-------D
|               |       +-------C
|               |       +-------E
|               |
|               +-------G
|
+-------L
        +-------J
        |       +-------I
        |       +-------K
        |
        +-------M
                +-------N
```

Figure 2: *Printing an unbalanced tree*

look like the following example:

```
d
+-------b
|       +-------a
|       +-------c
+-------f
        +-------e
        +-------g
```

(there should be a blank line above the *f* node). The code to do this is on lines 195—199. A blank line is printed if the current node is a right descendant and has no children. In the above example, this will happen at nodes *c* and *g*. The mirror-image issue is not a problem in a preorder traversal because you usually want to read down the columns (as in the above example). If you applied the same mirror-image reversal to the preorder traversal that you used in the in-order traversal, nodes *a* and *c* (and nodes *e* and *g*) would be transposed incorrectly.

The bit-map routines (*setbit( ), testbit( )*, and *makebitmap( )*) originally appeared in this column more than a

```
                              A
                    B
          D                   C

                              E
                    F
H                             G

                              I
                    J
          L                   K

                              M
                    N
                              O
```

Figure 3: *Output from* sinorder( )

```
Lines at this level correspond to this
bit in the map:

                Bit 1   Bit 2   Bit 3
                  .       .       .
                  .       .       +--- R
                  .       +---P---+
                  .       |   .   +--- O
                  +---N---+   .       .
                  |       .   +---M    .
              L---+   .               .
              .   |           +---K
                .                     .
              .   +---J---+   .       .
              .       .   +---I       .
              .       .       .       .
Node is
at depth:     0       1       2       3
```

Figure 4: *Relationships between bits and lines*

(continued from page 20)

year ago. They're reproduced in Listing Two, page 71.

Nonrecursive Tree Traversal

One of the problems with most traversal algorithms is that they're recursive. If a tree degrades to a linked list, you'll need as many recursion levels as there are nodes in the tree. Each recursion level requires its own stack frame, so assuming a 10-byte stack frame, you'll need 1,000 bytes of stack to traverse a degraded, 100-node tree. This can cause problems if huge amounts of stack aren't available, so occasionally it's useful to give up the simplicity of a recursive algorithm for an iterative one.

It's easy to search for a node or insert a node into a tree in a nonrecursive way because you never have to remember where you came from. That is, you only have to descend to the proper node and never have to go back up again. A nonrecursive search-and-insert function is given on lines 30–66 of Listing One.

Nonrecursive traversal is a harder problem because you have to go backward. In a recursive traversal routine, the pointer to the previous node is stored on the run-time stack, as part of the current subroutine's stack frame. That is, the pointer to the current node is passed to a recursive subroutine as a parameter. That parameter won't be modified by subsequent recursive calls because it's stored on the stack by each successive recursive call.

In an iterative traversal, you don't have the luxury of a run-time stack. One solution to this problem is to maintain your own stack of pointers to previous nodes as a static data structure, thereby moving the previous-node information from the run-time stack to the static data area. This method wastes space, though, because you already have a convenient place to store the previous-node pointers—in the tree itself. As you descend the tree, you can reverse the pointer that you just used to get to the current node so that it now points back up to the previous node. As you ascend back up, you reverse the pointer again. The basic traversal algorithm is shown in Table 1, below.

Deciding in which direction to go when you're at any given node is a problem. You'll go through every node three times— once on the way down, once again as you ascend from the left, and a third time as you ascend from the right. The problem is resolved by "marking" a node after you've printed the left subtree but before you descend right (that is, the second time you visit it). This way, when you come back up from the right, you can look at the mark and decide to ascend rather than go right again. Only one bit is needed to mark a node, so because my nodes have an ASCII string as one of their fields, I set the high bit of the first character in the string to mark a node. Macros to set, clear, and test this bit are on lines 23–25 of Listing One. If an extra bit isn't available, you can always add a tag field to the *LEAF* structure, but it seemed like a waste of memory to do that here.

The routine *lr_trav( )* (on lines 245–301 of Listing One) is a straightforward implementation of the algorithm in Table 4. It does an in-order traversal. Comments are inserted in the code to show where preorder or postorder visits should go (remove the in-order visits in these cases). The *pres* variable points at the node currently being visited. The *prev* variable points at the present node's parent. *Next* is just a convenient place to put things as you reverse pointers.

There are other nonrecursive traversal algorithms (the most interesting is the Robson traversal, which doesn't need to mark nodes), but they're all more complicated to implement than a simple link-reversal algorithm. If you're interested in these other methods, look at Thomas A. Standish's book *Data Structure Techniques* (Reading, Mass.: Addison-Wesley, 1980), 74–83.

```
do forever
    if( present node is marked )
        Clear mark.
    else
        while( there's a left child )
            Visit present node if doing preorder traversal.
            Go left.
        Visit pres node if in-order or preorder traversal.
    Visit present node if postorder traversal.
    if( no previous node )
        break
    if( previous node is marked )
        Go up from a right child.
    else
        Go up from a left child.
        Visit present node if in-order traversal.
        Mark the present node.
        Go right.
```

Table 1: *A link-reversal tree-traversal algorithm*

Microsoft C, Version 3.0

To no one's surprise, I received a call from Microsoft soon after my review of its compiler was published in the March 1986 C Chest. The company was (perhaps justifiably) miffed, so I agreed to give it equal time in the column—fair is fair. Sandra Jacobson (product manager, Systems Languages) at Microsoft sent me the following letter in response to that review:

"We appreciate Allen Holub's comments regarding the Microsoft C compiler. We try to improve each version of the compiler through comments and recommendations from users and reviewers.

"Although we agree with many of the comments made by Allen, as they have also been recommended to us by other users of Microsoft C, we still disagree with some major points. The following comments and recommendations have been incorporated into Version 4.0 of the compiler:

1. We have improved the internal error messages that are displayed. We have also improved on the error recovery in the parser.
2. The command-line interface has

HOW PEOPLE WITH COMMON INTERESTS FIND AN INTERESTING COMMON GROUND.

Presenting CompuServe Forums. Where people from all over get together, without even leaving home.

Now thanks to CompuServe Forums, computer owners are sharing common interests by talking to each other through their computer keyboards. ***Software users, computer enthusiasts, ham operators, french cooks, fire fighters, science fiction lovers and other special interest groups*** are already in touch, online.

Because when you subscribe to CompuServe, you're able to reach people who want to talk about the things you do. As many people as you like. For as long as you like. Whenever you wish.

Join a conversation already in progress or start one on your own. Ask questions. And get answers.

All it takes is a modem, most any personal computer and CompuServe.

Forum members across the country are as close as a local phone call.

You can go online with just a local call in most major metropolitan areas. And normal usage fees for weekday nights and weekends are just 10¢ a minute.

Of special interest to all Forum participants is software that's FREE for the taking.

Public domain software. For all sorts of activities, from games to business programs. And it's just as easy to copy a piece of software as it is to participate in a Forum.

Become a CompuServe subscriber and get a $25 Usage Credit to start you off.

Becoming a subscriber is as easy as contacting your local computer dealer. Or you can call us and order direct. Suggested retail price is $39.95.

And if you'd want more information about CompuServe, we'll be happy to send you a free brochure. Because with all that CompuServe offers—we think it's in your best interest.

CompuServe®

Information Services, P.O. Box 20212,
5000 Arlington Centre Blvd., Columbus, OH 43220
800-848-8199
In Ohio, call 614-457-0802

An H&R Block Company

Circle **no. 237** on reader service card.

C CHEST

(continued from page 20)

been improved to correct the 127-byte limit for flags passed to each of the compiler passes from the driver. There is still a 127-byte limit on invoking the compiler from the command line. We have also improved on the error recovery in the parser.

3. The internal error and problems with the *spawn* function have been fixed.

4. Because of requests from users of Microsoft C, we have included the C start-up code with Version 4.0 of the C compiler. It is considerably more than 100 lines of code, and these routines should be used only by experienced C programmers who have a complete understanding of C and MS-DOS. This will considerably help people who want to create ROMable code.

5. The documentation on the extensions *near* and *far* has been improved in Version 4.0.

"We feel that some of the problems that Allen discussed were based on incorrect assumptions:

• The differences between 'the way a Microsoft library routine works and . . . the equivalent Unix routine' are specified in Appendix B of the *Library Reference Manual*: 'A Common Library for XENIX and MS-DOS.' *[I missed it, sorry.—Allen]*

• In reference to '. . . *strncpy* pads out the string with nulls to the maximum count—why?', our implementation of *strncpy* is compatible with the Unix System V definition of *strncpy* as well as the definition in the developing ANSI standard. Both require padding of the target string. *[I still wonder why, but the problem isn't Microsoft's.—Allen]*

• With respect to Allen's concern about *#endif*: '. . . it ignores the last line of a file if it isn't terminated with a new line. An *#endif* without a *CR* gives an "unexpected end of file" message.' The developing ANSI standard (which we're following closely) specifies that a new-line character (*CR-LF*) is required as the last character of every source file that is referenced via an *#include* statement. Because any file may be referenced in this way, Microsoft C does not differentiate between included source and main source. *[Oh come off it. How hard can it be to check for* EOF *as well as* CR-LF*?—Allen]*

"We try to provide products that are worthy of use by a wide variety of users, both in-house and outside Microsoft. We encourage a strong dialogue between our users and our product-support group. Allen claims that we don't support our product, yet he told us that he didn't contact Microsoft to verify that his problems were valid or to report them. *[Mea culpa.—Allen]* The best way for a user of Microsoft C to make sure that corrections and recommendations to the compiler are made is to contact Microsoft Technical Support at (206) 882-8089. Problem reports can also be sent to us at our new address: Microsoft Corp., Technical Support, 16011 N.E. 36th Way, P.O. Box 97017, Redmond, WA 98073-9717. For compiler errors, we suggest that the localized problem be sent in on a disk."

I'm tempted not to carry this discussion any farther, but some of Ms.

If you're tired of waiting, you're using the wrong file manager.

Be fast. Btrieve.®

If batch jobs and reports turn waiting time to nighttime, then wake up! You're using the wrong file manager.

Btrieve® file management hates waiting as much as you do. It's written in assembly language especially for the IBM PC. And based on b-tree file indexing, with automatic balancing for access speed that won't degrade as your database grows. With Btrieve, your applications always run fast. So you'll be out the door faster.

The standard for networking. Btrieve/N (network version) sets the standard for the industry's most popular LANs and multi-user systems. Btrieve/N offers *safe* network file management that coordinates simultaneous updates and prevents lost data.

Automatic file recovery. Btrieve provides automatic file recovery after a system crash. Your Btrieve data always comes back intact.

Fully-relational data management. SoftCraft's entire family of products gives you a complete, fully-relational database management system. Rtrieve™ adds report writing capabilities. Xtrieve™ speeds users through database queries with interactive menus.

For professional programmers. Btrieve is the fast, reliable answer for all your application development in BASIC, Pascal, COBOL, C, FORTRAN and APL. With Btrieve, you can develop better applications faster. And know they'll run—*fast.*

P.O. Box #917 Austin, Texas 78766
(512) 346-8380 Telex 358 200

Suggested retail prices: Btrieve, $245; Btrieve/N, $595; Xtrieve, $245; Xtrieve/N, $595; Rtrieve, $145; Rtrieve/N, $345. Requires PC-DOS or MS-DOS 1.X, 2.X, or 3.X. NO ROYALTIES.

Circle **no. 113** on reader service card.

The Full-Service Source For Programming Software

Knowing software is an art in itself. It takes time and dedication. At Lifeboat, we're committed to more than just selling the best software. We know our products and support them. And we learn about you, our customers, giving you the complete solution to your programming needs.

LANGUAGES

Lattice C New 3.0 Version The best selling C Compiler has been upgraded to give you more functions and features. Lattice C 3.0 contains 200 new library functions, better code generation, support for new data types (void, enum, unsigned char, unsigned long), support for the 80186/80286 instruction set and the ability to generate in-line 8087/80287 instructions. Lattice C is the C compiler for professional developers.

RUN/C—The C Interpreter Upgraded Version Learn C the natural way with RUN/C. The user interface is similar to BASIC with easy familiar commands. The new 2.0 version of RUN/C comes with a full-screen editor and other enhancements.

RUN/C Professional New RUN/C Professional "...is the overall best choice in a C interpreter." *PC Tech Journal* (5/86). All RUN/C's capabilities plus powerful features for program development. Load your favorite object libraries with RUN/C Professional. Contains a full-screen editor and source code debugging facilities.

BetterBASIC New Version Now you can program in BASIC and use the full memory of your PC, create structured programs using functions and procedures, make your own library modules and more. Now compatible with Microsoft BASIC.

LANGUAGE UTILITIES

Periscope I & II The symbolic debugger that can debug anywhere within a program at anytime with a simple press of a button! Periscope's breakout button will interrupt infinite loops, gain freedom from a locked up keyboard or help you find out where that slow running program is spending all its time. Periscope covers every conceivable type of breakpoint, including breaks on reads and writes to ranges of memory. Source code and line numbers for popular high level languages are displayed. Periscope I additionally features a protected memory board which shields the debugger from runaway programs.

Plink86 Plus An overlay linkage editor for linking 8086/8088 object modules. Contains new features for memory caching, library allocation, file merging and overlay reloading.

Pfix86 Plus A symbolic and source level advanced debugger for programming professionals.

C-SPRITE Symbolic debugging at both source level and machine level. Monitor your program by setting breakpoints, examining registers and variables, or single-stepping through code.

EDITORS

FirsTime Speed-up program development with an easy-to-use syntax checking editor for C or Pascal. Detects syntax errors, undefined variables and mismatched type assignments. Use function key menus to generate statements or enter your code directly. Also includes a program formatter and unique program development aids. FirsTime key assignments can be reconfigured. Easy-to-learn with on-line help available.

VEDIT Plus A full-screen text editor for program development and word processing. It contains powerful features including use of macros, on-line help facility, paragraph formatting, and file comparison.

Pmate The programmer's editor with an extensive macro command language. Compile in the background while you continue to edit files.

EMACS Customizable editor including windowing, multi-tasking and special modes for C and Pascal.

FUNCTIONS

The Greenleaf Comm Library A library of over 120 communication routines. Contains functions to create interrupt driven routines or perform direct I/O to multiple Comm Ports. Its strengths are in asynchronous communications, interrupt mode, modem control, XMODEM, XON/XOFF and flow control. Interfaces with Lattice, Microsoft, Wizard, Desmet, CI86, Aztec and Mark Williams C compilers.

PforCe New A library of 400 plus functions for interrupt driven communications, background tasks, string/table parsing, data base manipulation, DOS and BIOS functions and more. Superbly written and documented software with complete source code is included.

The Greenleaf Functions A mature library of over 200 functions. Version 3.0 offers all new indexed documentation, with an abundance of examples. Source code included.

BASIC_C Speed-up development time with this BASIC to C library. Designed for the BASIC programmer who is moving over to C. This library act as a bridge to C providing many of the capabilities of the BASIC language.

Essential C Utility Library Over 300 functions, with special attention given to screen handling, windows and business graphics. Source code is included.

GRAPHICS, WINDOWING and SCREEN DESIGN

GSS*CGI GRAPHICS New The GSS Computer Graphics Interface is designed for creating high performance graphics-based applications. GSS*CGI speeds up application development and provides compatibility with a wide range of peripherals. It's the only CGI implementation that provides true device-independence for both raster and vector graphics. Products in the GSS GRAPHICS line include: the GSS*CGI Graphics Development Toolkit, the GSS Kernel System; the GSS Plotting System; and the GSS*CGI Metafile Interpreter.

Essential Graphics New A brand new graphics library for C programmers with the emphasis placed on ease of use and portability. No royalties.

Halo Library of over 200 graphics functions, supporting all of the popular graphics boards.

Windows for Data New Provides C programmers with complete control over screen display and data entry, within an easy-to-use windowing system. Combines a comprehensive library of window functions with an integrated set of versatile data-entry functions. Features include one-step data entry, multiple data types, Lotus-style menu design, full-featured field editing, pop-up entry windows, and much more.

Panel A powerful tool for interactive screen design.

For more information on these and other products in our complete line call:

1-800-847-7078
NY: 914-332-1875

The Full-Service Source for Programming Software.

LIFEBOAT

INTERNATIONAL SALES OFFICES

Australia:
Fagan Microprocessor Assoc
Phone: (61) 3699-9899
Canada: Scantel Systems
Phone: (416) 449-9252
England: Grey Matter, Ltd.
Phone: (44) 364-53499
Italy: Lifeboat Assoc., S.p.A.
Phone: (02) 656-841
Japan: Lifeboat Japan
Phone: (03) 293-4711
Spain: Micronet, S.A.
Phone: (34) 1-457-5056
The Netherlands:
GIGA Computer Products
Phone: (31) 10-771846
W. Germany:
MEMA Computer GmbH
Phone: (49) (069)34 72 26

Lifeboat
55 South Broadway
Tarrytown, NY 10591

The names of products listed are generally the trademaks of the sources of the products.
©1986 Lifeboat Associates

Circle **no. 118** on reader service card.

C CHEST
(continued from page 22)

Jacobson's points deserve comment. Though my remarks both here and in the original review are critical of the compiler (criticism is, after all, the point of a critical review), I do think the Microsoft C compiler is a good product (at least when contrasted with other available compilers). I use it myself. It could be a better product, though, and I'm hoping that a public discussion of its faults will spur Microsoft into making a few essential changes. All too often, new products are greeted with encomiums rather than real analyses, and I think that honest critique from a working programmer's perspective is important. Nobody benefits if I don't talk about problems I find in a product just because the manufacturer of that product claims that they'll be fixed in the next version. Even if the technical-support group can tell me about a bug over the phone, the bug still exists, and not everyone can spend lots of time on the phone to Washington. I don't want to antagonize Microsoft. What I do want is a better compiler, and I honestly believe that public discussion is the only way to bring this about.

I would love to say that the Microsoft C compiler is the greatest thing since peanut butter, but I couldn't say that about Version 3.0 without my nose growing several inches. I've not received a review copy of Version 4.0 yet, but it sounds as though most of the major faults in the compiler (at least the bugs and the error-recovery problems) have been fixed. I look forward to getting a copy.

One major problem that isn't addressed in the letter is the poor overall quality of the *User's Guide.* To my mind a user's guide for a C compiler shouldn't also try to teach you how to use DOS—that's what the DOS manual is for. I hope that Microsoft will seriously consider rewriting it for experienced programmers who can read above a sixth-grade level. As for technical support, at $400 for the compiler, I think Microsoft should, at the very least, send out an occasional newsletter listing all known bugs and work-arounds for them (I've suggested this to Microsoft, and it is considering it). If the firm doesn't tell you about a known bug, then the bug is undocumented, whether or not that bug was discovered after the documentation was printed. By the same token, the technical-support phone number should be a toll-free number (it isn't). Having to pay peak-hour phone rates to find out about a bug that should have been reported in a newsletter is just adding insult to injury. Finally, I don't consider sending in a bug report and getting an answer at some indeterminate future date to be adequate support. I want to call and get an answer right then. If the normal technical-support person can't answer a question, I want to be connected to someone who can. It seems reasonable to have to pay extra for this kind of support, but I think it should be available. (Microsoft is considering doing this, too.)

Microsoft, of course, is not alone in providing inadequate (I think) technical support, and in all fairness, it seems to be interested in improving its support process. I think there's a technical-support problem in the industry as a whole. A "these guys are just hobbyists, so what do they need source code or schematics for?" attitude seems to prevail. My fond hope is that Microsoft, which in my experience is a major offender in this department, will have a change of heart and lead the way for better technical support in the industry overall. All hardware should be shipped with schematics and a complete technical description. Period. All software documentation should explain low-level internals in depth, and source code should always be available if you need it. Period.

Lattice C, Version 3.0

I received a copy of Version 3 of the Lattice C compiler (3.0F to be exact) a few months ago, and I've finally had a chance to look at it. This version represents a significant improvement over previous versions. Function prototyping (strong type checking of subroutine arguments) has been added, and the compiler has been made more Unix-compatible overall (for example, *unsigned* is now a modifier rather than a type). The library has also been expanded considerably. If you have an earlier version of the compiler, I'd definitely recommend an upgrade, even if you have to pay for it.

To test the compiler, I recompiled all the routines in the /util program package that *DDJ*'s currently distributing. I found both good and bad things. The code-size problem has been addressed. The .exe files are now considerably smaller than they used to be. On the average, Lattice executables are only 7 percent larger than the Microsoft executables (generated from the same source code). Some of the Lattice executables are literally half the size of the Microsoft versions, however.

The compiler's error messages are pretty good, and there is now some *lint*-like support. For example:

```
foo( )
{
    int     i;
    return;
}
```

gives the error message "FOO.c 5 Warning 93: no reference to identifier 'i'." Like *lint*, however, this kind of error checking can generate needless noise. For example:

```
foo( )
{
    int i;
    int a=1;

    for( a; a; i++ )
        i = 6;
}
```

generated "FOO.c 6 Warning 94: uninitialized auto variable 'i'." I'd like to see a way to disable these *lint*-like error messages (with the exception of the function prototype warnings).

The Lattice compiler has several subroutines [*dosint( )*, for example] that have the same names, but different calling conventions, from those of the Microsoft compiler. This situation came about because they were originally the same compiler, but I wish one or the other of the manufacturers would change its subroutine names. Microsoft provides an include file with *#defines* that takes care of most of the changes. Lattice should provide a similar file to go in the other direction.

There are a few Unix I/O library functions missing from the library, though there are often functional

C Programmers! First database written exclusively for C is also royalty free

"If you are looking for a sophisticated C Programmer's Database, db_VISTA™ is it..."
Dave Schmitt, President, Lattice, Inc.

Designed exclusively for C, db_VISTA™ is a royalty-free programmer's DBMS. Both single and multi-user versions let you take full advantage of C, through ease of use, portability and efficiency.

Written in C for C Programmers

All functions use C conventions so you will find db_VISTA easy to learn. db_VISTA operates on most popular computers, and because it is written in C it can easily be ported to most computers.

Royalty-Free, You only pay once

Whether you're developing applications for a few customers, or for thousands, the price of db_VISTA is the same. If you are currently paying royalties for a competitor's database, consider switching to db_VISTA and say goodbye to royalties. To help you make the change over to db_VISTA, ASCII file transfer utilities are included. dBASE file transfer utilities are available as an option.

More from your database applications with source code

Source code includes all db_ VISTA libraries and utilities.
1. Recompile our run-time libraries utilizing non-standard compiler options.
2. Create a debugging library including a function traceback by activating preprocessor commands embedded in the source code.

Multi-user and LAN capability

Information often needs to be shared. db_VISTA has multi-user capability and supports simultaneous users in either multi-tasking or local area networking environments, allowing the same C applications to run under UNIX and MS-DOS.

Faster execution without data redundancy

Less data redundancy means reducing disk storage requirements and maximizing data access performance. A customer evaluating a leading competitor's product prior to purchasing db_VISTA benchmarked db_VISTA's retrieval time to be 276% faster than a leading competitor.

Complete documentation included

User manual contains 193 pages, 8 diagrams, 10 tables, appendices, an extensive index, plus a database application example. 9 chapters with complete instructions.

Introducing db_QUERY™

With db_QUERY you can ask more of your database. db_QUERY is a linkable, SQL-based ad hoc query and report writing facility. It's also royalty-free.

30 day Money-Back Guarantee

We wish to give you the opportunity to try db_VISTA for 30 days in your development environment and if not satisfied return it for a full refund.

SAVE $50 TO $100! PURCHASE db_VISTA™ BY AUGUST 31, 1986 and, RECEIVE UP TO $100 REBATE*

Price Schedule

	db_VISTA	Rebate
Single-user	$195	
Single-user with Source	$495	$ 50
Multi-user	$495	$ 50
Multi-user with Source	$990	$100

Free 90 days application development support
All software Not Copy Protected

Call Toll Free Today and Learn How To Receive Up To $100

To order or for information, call TOLL FREE 1-800-843-3313, at the tone touch 700-992 or call 206-747-5570.
VISA and MASTERCARD Accepted

Read what others say about db_VISTA..

"If you are looking for a sophisticated C programmers database, db_VISTA is it. In either a single or multi-user environment, db_VISTA lets you easily build complex databases with many interconnected record types. The multi-user implementation handles data efficiently with a LAN and Raima's customer support and documentation is excellent. Source code availability and a royalty-free run-time is a big plus."
Dave Schmitt, President Lattice, Inc.

"Not 'yet another user-friendly database', it is a DBMS aimed at the technical C programmer instead of the non-technical end-user".
Hal Schoolcraft, Data Based Advisor March, 1985

"On the whole, I have found db_VISTA easy to use, very fast with a key find, and powerful enough for any DBMS use I can imagine on a microcomputer".
Michael Wilson, Computer Language September, 1985

db_VISTA Version 2.11 Database Mangement System for C

Database Record and File Sizes
- Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215
- No limit on number of records or set types
- Maximum file size limited only by available disk storage
- Maximum of 255 index and data files

Keys and Sets
- Key length maximum 246 bytes
- No limit on maximum number of key fields per record - any or all fields may be keys with the option of making each key unique or duplicate
- No limit on maximum number of fields per record, sets per database, or sort fields per set
- No limit on maximum number of member record types per set

Utilities
- Database definition language processor
- Interactive database access utility
- Database consistency check utility
- Database initialization utility
- Multi-user file locks clear utility
- Key file build utility
- Data field alignment check utility
- Database dictionary print utility
- Key file dump utility
- ASCII file import and export utility

Features
- Multi-user support allows flexibility to run on a local area network
- File structure is based on the B-tree indexing method and the network database model
- Run-time size, variable - will run in as little as 64K, recommended RAM size is 256K
- Transaction processing assures multi-user database consistency
- File locking support provides read and write locks on shared databases
- SQL-based db_QUERY is linkable
- File transfer utilities included for ASCII, dBASE optional

Operating System & Compiler Support
- Operating system's MS-DOS, PC-DOS, Unix, Xenix, Macintosh & Amiga
- C compiler's Lattice, Microsoft, DeSmet, Aztec, Computer Innovations, Xenix and Unix

Independent Benchmark Results
Eleven key retrieval tests on sequentially and randomly created key files. Benchmark procedure adapted from "Benchmarking Database Systems: A Systematic Approach" by Bitton, DeWitt, and Turbyfill, December, 1983

Total Retrieval Time of 11 Tests	
db_VISTA	:671.24
Leading competitor	:1,856.43

RAIMA™ CORPORATION
12201 S.E. Tenth Street
Bellevue, WA 98005 USA
(206) 747-5570
Telex: 9103330300 BCN RIVERTON

1 (800) 843-3313
at the tone touch 700-992

MasterCard VISA

* Limited offer available to end-user purchases directly from Raima Corporation.

C CHEST
(continued from page 24)

equivalents in the library. Lattice doesn't have a *stat( )* system call, for example. It has several functions that give you the same information as would *stat* [for example, *getfa( )* and *access( )*]. Nonetheless, I'd like to see a Unix-compatible function, too.

I also found a couple of trivial but annoying bugs—for example, the environment string vector array [pointed to by the *envp* argument to *main( )*] isn't terminated in the right place (there's one too many entries, and the last one is garbage).

A rather weird bug showed up when I tried to redirect-append (>>) the output from Lattice-compiled programs from the Microsoft-compiled shell. The Lattice-compiled programs overwrote the file rather than adding text to the end of it. The same programs had no problem when running under command.com, and they also worked fine with simple redirection (>) with the shell. I'm not blaming Lattice for this one [seeing as Microsoft C has *spawn( )* problems, I suspect the problem lies there], but the problem did arise.

Finally, Lattice told me that my version of the compiler had bugs in the in-line 8087 code generation. Though these particular problems should be fixed by the time you see this column, I haven't received an update so I can't verify this.

Perhaps the biggest potential problem with the compiler is the documentation. Lattice has returned to the insanity of a manual augmented with a *Technical Bulletin*. The manual I received was the same one that was shipped with Version 2.15. It is supplemented with a *Technical Bulletin* that is almost the same size as the manual. At least there's an index that references both volumes, but it's still annoying to have to go back and forth—I always seem to pick up the wrong volume. The *Technical Bulletin* lists all the library routines in strict alphabetical order, which is good. The manual itself does not, however—routines are grouped by function, another annoyance when you're trying to find something. Moreover, the bulletin doesn't have a functional index (one that groups library routines by function and then provides a capsule description of each routine). It has an alphabetical index with capsule descriptions, but this isn't much use if you know what you want to do but don't know the name of the routine that does it. A note came with the compiler saying that the company is working on a single, integrated manual that will be sent to all registered users at no charge, but I haven't seen this new manual yet.

Another problem is the amount of work you have to do to compile a program. The Lattice compiler comes with a horde of batch files and libraries. Even with these batch files, compiling is not a one-step process because you have to juggle parameters to the linker too. It's up to you to remember what libraries to link. One of the things I really like about the Microsoft compiler is the one-step cl driver program. It's an almost exact look-alike of the Unix cc driver and takes most of the pain out of compiling. It figures out what libraries to link and even invokes the linker for you. The Lattice compiler could benefit immeasurably from the inclusion of a similar driver program.

One final problem: The Lattice compiler still can't generate assembly-language source files directly. Lattice provides you with a program that disassembles object modules, but I've never liked this approach. It's just too awkward to use. With any new release of a compiler, it's critical to be able to see the code that the compiler generates. That's the only way to tell if an error is yours or if it is a bug in the compiler itself. Object-module disassembly makes it unnecessarily difficult to get at the assembly-language source. It's also another program that could potentially introduce bugs into acceptable object code (though I've never seen this happen, I worry about it).

In conclusion, I think this version of the compiler is significantly better than previous versions. Lattice is back in the running, at least in terms of features and code size. The compiler, in spite of all its improvements, is not a clear winner, though. Some of the problems (such as the 8087 bugs) are the result of a too-early release date. Similarly, the library documentation in its present form is not really acceptable (though it's a lot better than some manufacturers' documentation). The compiler is also harder to use than I'd like, and the library is not as Unix-compatible as I'd like. On the other hand, Lattice not only gives you root-module sources but will also sell you the source for the rest of the I/O library. (I've heard talk of Microsoft doing the latter, but I haven't seen an official announcement yet. Microsoft is releasing the root module with Version 4.0. Manx gives you all the sources if you buy its professional package.) Various sources for the error-processing routines are also included with the Lattice compiler package. So this, too, is a good product. I like it, but there's plenty of room for improvement.

Coming Attractions

Next month I'll continue looking at trees. I'll refine the in-order graphic traversal routine a little further, and I'll look in depth at AVL balanced trees. AVL trees are guaranteed to be almost perfectly balanced (the imbalance is at most one level), so they're very useful when you want a best-case access time and don't care if it takes a little longer to create the tree. I'll also look at the problem of deleting a node from a binary tree.

Availability

The shell is available from *DDJ* (see advertisement on page 78). All the code published this month is available both on CompuServe (type go ddjforum) and, for $25, on an IBM PC-compatible disk from Software Engineering Consultants, P.O. Box 5679, Berkeley, CA 94705. The tree routines that I'll look at next month are on the same disk.

DDJ

(Listings begin on page 68.)

Vote for your favorite feature/article.
Circle Reader Service **No. 2.**

DR. JACK PURDUM

Get the proven product from the man who wrote the books on "C".

THE FIRST PROFESSIONAL 'C' COMPILER FOR UNDER $60.00

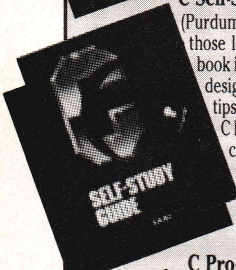
Limited Time Offer **FREE** CED TEXT EDITOR WITH "BUILT-IN" FUNCTION HELP.

C Programming Guide

After reading the 1st edition, Jerry Pournelle (**BYTE** Magazine) said: "I recommend this book...Read it *before* trying to tackle Kernighan and Ritchie." The second edition expands this best seller and walks you through the C language in an easy-to-understand manner. Many of the error messages include references to this book making it a perfect companion to Eco-C88 for those just starting out with C.

$20

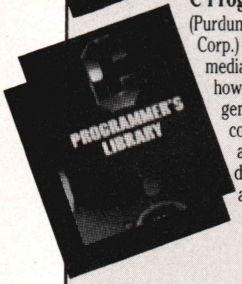
C Self-Study Guide



(Purdum, Que Corp.) Designed for those learning C on their own. The book is filled with questions-answers designed to illustrate many of the tips, traps, and techniques of the C language. Although written to complement the Guide, it may be used with any introductory text on C.

$17

C Programmer's Library



(Purdum, Leslie, Stegemoller, Que Corp.) This best seller is an intermediate text designed to teach you how to write library functions in a generalized fashion. The book covers many advanced C topics and contains many useful additions to your library including a complete ISAM file handler.

$22

CED Program Editor

CED now supports on-line function help. If you've forgotten how to use a standard library function, just type in the name of the function and CED gives you a brief summary, including function arguments. CED is a full screen editor with auto-flagging of source code errors, multiple windows, macros, and is fully configurable to suit your needs. You can edit, compile, link, and execute DOS commands from within the editor. Perfect for use with Eco-C88. For IBM PC, AT and look-alikes.

$29.95

C Source for Standard Library

Contains all of the source code for the library functions that are distributed with Eco-C88, excluding the transcendentals and functions written in assembler.

$10 ($20 if not with order)

Developer's Library

Contains all the source code for the standard library, including transcendental and assembler functions. Available with compiler purchase only.

$25 ($50 if not with order)

Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

NOTHING IN THIS PRICE RANGE EVEN COMES CLOSE.

"Ecosoft's Eco-C is going to turn some major heads."

"Eco-C is the first compiler reviewed that has clearly begun implementing the coming ANSI standard. Eco-C supports prototyping."

"Eco-C performed well on all the benchmarks, generating code that was quite comparable to that of compilers 10 times as costly."

from:
Christopher Skelly
Computer Language,
Feb., 1986

"The driver program is another strength: it compiles and links a list of files and provides a simple MAKE capability."

"This compiler does handle syntax errors much better than average—no avalanche of spurious messages here."

from:
William Hunt
PC Tech Journal,
Jan., 1986

"Eco-C88 is a high-quality package... comparable to systems costing much more."

"Eco-C88 is one of the fastest..."

"It is convenient to use, works well, and produces acceptably compact and fast programs."

from:
Dr. David Clark
Byte, Jan., 1986

Minimum System Requirements:

To use Eco-C88 Release 3.0, you must have:

1. An IBM PC, XT, or AT-compatible computer with monitor.
2. 256K or more memory.
3. Two 360K disk drives, or a hard disk.
4. PC or MSDOS 2.1 or later to include the MSDOS linker.

Eco-C88, mini-make, memfiles and CED are trademarks of Ecosoft Inc.
IBM is a trademark of International Business Machines.
UNIX is a trademark of Bell Labs.
MSDOS and MASM are trademarks of Microsoft.

Full-featured C compiler. Supports all C features, data types (except bit fields), and operators.

New Language Enhancements. You also get prototyping, enum and void data types, plus structure passing and assignment.

Tiered Error Checking. All syntax errors are automatically flagged, but you can select the level of "link-like" semantic error checking you want.

Complete Standard Library. Over 200 functions, many of which are System V compatible for greater source code portability.

Screen and Memory Functions. Now you can write programs that use color, cursor addressing, even ones that let you design your own graphics functions. You also get memfiles™ that allow you to access memory outside the normal data segment as a file.

8087 and 80287 Support. If you have one of these math chips, your programs will take immediate advantage of it. If you don't have one, the code automatically switches to software floating point.

Full Screen Editor. The CED editor is a full screen editor with multiple windows, macro commands, on-line function help, plus a full set of editing commands. (Requires a true IBM PC compatible.) You can edit, compile, link, and execute programs from the editor which greatly reduces development time.

Includes a cc and mini-make™. The UNIX-like cc makes compiling programs a snap. You can run cc from within the CED editor.

ASM or OBJ Output. You can select assembler or relocatable output from the compiler. Both are MASM compatible and ready for use with the MSDOS linker to produce EXE files.

	Eco-C88	Lattice	Computer Inn. C86	Microsoft	Mark Williams
sieve	12	11	13	11	12
fib	43	58	46	109	–
deref	14	13	–	10	11
matrix	22	29	27	28	29

Computer Language, Feb., 1985, p. 79. Reproduced with permission.

Orders only: **1-800-952-0472**

ORDER FORM CLIP & MAIL TO: Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

- ☐ C Compiler $59.95 ____
- ☐ C Programming Guide $20.00 ____
- ☐ C Self-Study Guide $17.00 ____
- ☐ C Programmer's Library $22.00 ____
- ☐ CED Program Editor $29.95 ____
- ☐ C Source for Standard Library $10.00 ($20.00 if not with order) ____
- ☐ Developer's Library $25.00 ($50.00 if not with order) ____

SHIPPING $4.00

TOTAL (IND. RES. ADD 5% TAX) ____

PAYMENT: ☐ VISA ☐ MC ☐ AE ☐ CHECK

CARD # ____ EXPIR. DATE ____

NAME ____

ADDRESS ____

CITY ____ STATE ____

ZIP ____ PHONE ____

ECOSOFT

Circle no. 89 on reader service card

Ada just moved into a smaller place.

We have some very good news for you.

You can now get a validated, full Ada® compiler for the IBM® PC AT. From the people who designed the Ada language. For just $3,000.

Which means you and your company can now program in Ada without tying up a big, expensive computer.

And you should program in Ada. And not just because the DoD says so.

alsys

1432 Main Street, Waltham, MA 02154

☐ Send me more information about the Alsys PC AT Ada compiler.
☐ Send me more information about Ada.

Name ______________________

Title ______________________

Company ______________________

Address ______________________

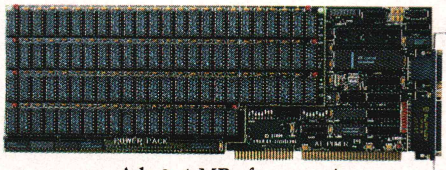
City ______________ State ________ Zip ________

Telephone ______________________

The DoD mandates Ada for their software principally because Ada is considerably easier and less expensive to maintain.

Does more reliable and easier to maintain code sound attractive to you? If not, just look at how your programmers are spending 80% of their time.

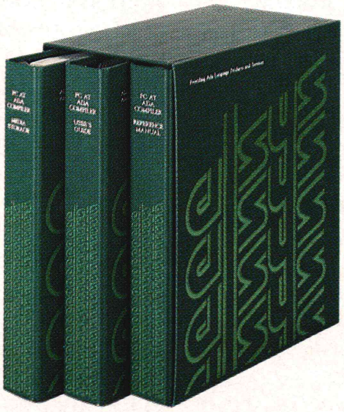
People who know Ada are calling it "the only logical language for the eighties and nineties." The point is, they're not thinking of Ada as "the DoD language." It's simply because Ada supports good, solid software engineering practice.


Ada & 4-MB of memory* for the price of 4-MB of memory.

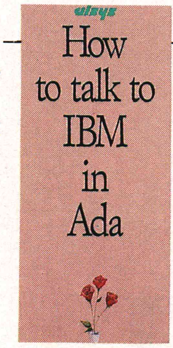
And now you can try full Ada programming for less than the cost of a two week training program.

The Alsys™ Ada compiler for the PC AT is not only validated, it's actually written in Ada. And produces code so efficient it executes faster than C or Pascal on tested benchmarks.

And if that's not enough, the Alsys PC AT Ada compiler runs in protected mode. So you can use the full amount of memory available to the PC AT. This means you can run a program using up to 16 megabytes of memory for code and data without worrying about DOS, overlays and all that stuff.



Lastly, this compiler comes with a 4-megabyte memory upgrade board. That, by itself, is worth the price of admission.



Knowledge is good. Especially when it's free.

At this point, we suspect you might be tempted to pull out a credit card and give us a call.

Or at least fill out the coupon.

Write: Alsys, Inc., 1432 Main Street, Waltham, MA 02154, U.S.A., Telephone: (617) 890-0030, Telex: 948536.

In France: Alsys, S.A., 29, Avenue de Versailles, 78170 La Celle St. Cloud, France, Telephone: 33(1)3918 12 44, Telex: 697569.

In England: Alsys, Ltd., Partridge Hse, Newtown Road, Henley-on-Thames, Oxon RG9 1EN, England, Telephone: 44 (491) 579090, Telex: 846508.

Ada® is a registered trademark of the U.S. Government (Ada Joint Program Office). IBM® and PC AT are registered trademarks of International Business Machines Corporation. Alsys™ is a trademark, service mark, and trade name of Alsys S.A. *Memory board manufactured by Profit Systems, Inc.

Circle **no. 273** on reader service card.

A Forth Standards Proposal:

Extended Control Structures

by George W. Shaw II

The structures proposed in this article cover almost every control structure ever proposed.

The control structures in the Forth 83 Standard leave something to be desired. This is less stressful than it seems as all Forth programmers or systems implementors simply write their own prescriptions for the ills that plague them most. Unfortunately, of those published, all fail to supply a general solution to the problems, few are written in a manner that would pass the Standards Team, and none of them are part of the current standard.

Many partial solutions have been made public, but a complete solution becomes riddled with new words or plagued by unclear or nonstandard syntax. Many efforts solve only limited problems or do not maintain compatibility with existing control structures. It seems clear that often the authors are not aware of the general problem, are able to solve only a specific fragment, or overgeneralize their solution at the expense of compatibility and clarity.[1]

The Standard

Table 1, page 31, lists the syntax of the standard control structures. They are fairly simple. The standard does not specify the implementation of standard words, only their behavior. The code for the control structures in most Forth implementations, however, is very close to that presented in Listing One, page 82. (For clarity, compiler security is omitted.) Note that the listing includes the System Extension Word Set compiler-layer words *>MARK*, *>RESOLVE*, *<MARK*, and *<RESOLVE*. The System Extension Word Set nucleus-layer words *BRANCH* and *?BRANCH* are assumed available, as well as the nonstandard run-time words *(DO)*, *(LEAVE)*, *(LOOP)*, and *(+LOOP)*.

© 1986 by George W. Shaw II, Shaw Laboratories Ltd., 24301 Southland Dr., Ste. 216, Hayward, CA 94545-3471

There's more than one way to strip a parity bit, and the standard control structure words are no exception. One implementation variant of interest is *LEAVE*, which is used to exit a *DO* loop. The implementation options were the first step in reaching the solution presented in this article.

The History

In Forth 79, *LEAVE* does not exit *DO* loops directly but typically adjusts the *DO* loop's parameters so that the next time *LOOP* or *+LOOP* executes the loop will terminate. With the new, better *DO*-loop operation specified in Forth 83, the old *LEAVE* behavior no longer works. *LEAVE* cannot adjust the *DO*-loop parameters so that *LOOP* or *+LOOP* can reliably determine if they should terminate. A different behavior of *LEAVE* is required. Hence, Forth 83 specifies that *LEAVE* exits the loop immediately.

That is fairly straightforward, right? Wrong. Because *LEAVE* must be conditionally executed to be useful (see Table 1), it must have the ability to exit from within any structure in which it might be nested, to the point just past the next *LOOP* or *+LOOP*. Because Forth is very structured, during compilation almost all branch and structure addresses are simply nested on the stack and resolved from the stack. Thus, compiling an "unstructured" branch to just after *LOOP* or *+LOOP* pierces the nested levels and must be handled differently. To further complicate things,

the standard specifies that multiple *LEAVE*s are allowed in a *DO* loop. One of two approaches is usually taken to solve the compilation problem.

In the first approach, the address just after *LOOP* or *+LOOP* is stored with the loop parameters and is accessed when *LEAVE* is executed. It is easy to compile because there is no branch address to resolve during compilation. This approach also uses minimal overall memory and actually saves memory with multiple *LEAVE*s compared to the second approach. Its limitations are that all *LEAVE*s branch to the same place (not a problem within the standard) and additional loop parameter space is required for each nesting level of a *DO* loop.

The second approach is to allow each compiled *LEAVE* its own branch address. (See Listing One.) It uses minimal overall memory when a single *LEAVE* exists in a *DO* loop (99 percent of the cases) and just slightly more memory (for each branch address) than the first approach when multiple *LEAVE*s exist in a *DO* loop. This approach also uses minimal loop parameter memory and allows each *LEAVE* its own unique destination, albeit nonstandard. It appears difficult to compile, but this is not the case. The branch addresses are simply maintained in a linked list, the *LEAVE-LIST*, which is resolved by *LOOP* or *+LOOP*. Linked lists are often needed elsewhere in the Forth system, so the overhead of the compiling words *>MARKLIST* and *>RESOLVESLIST* may be nonexistent. Once you accept the power vs. compilation-complexity trade-off, some very useful structures can be built.

LEAVE naturally appears somewhat unstructured, though it does comply with the structured programming rule it seems to break most flagrantly. Structured programming requires that each program module have a single entry point and a single exit point. In a *DO* loop, the entry point is at *DO*, and the exit point is just after *LOOP* or *+LOOP*. *LEAVE* branches to just after *LOOP* or *+LOOP*. In a puff of logic, *LEAVE* becomes somewhat structured.

Common Extensions

Most Forth implementations extend the standard control structures a bit, frequently in the area of *BEGIN* loops. These are fairly simple and obvious extensions that, despite these facts, are nonstandard. If you replace the corresponding standard style code in Listing One with the code in Listing Two, page 82, you have typical extensions. These upgrades are significant, but no great shakes.

The behaviors of the typical extensions are not part of the standard but are compatible with it. They allow the syntax additions shown in Table 2, right. Code for the *CASE* statement listed in Table 2 is not supplied because it is probably not a candidate for standardization because its usefulness is limited. *OF* only allows checking for the equality case on a 16-bit value. If the values are equal, the code following *OF* is executed; otherwise execution continues after *ENDOF*.

Unresolved Problems

The extensions are useful and implementation is fairly simple, so what is the problem? (See Table 3, right.) All the problems listed stem from the desire to determine the exit trail of a loop. A programmer trying to write standard code is currently required to float a flag or value on the stack (heaven forbid you should use a variable) to indicate the exit trail. Because you cannot exit where you need to, you must retest after the loop for what you already knew when you were in the loop but had to lose.

The first three problems apply to standard control structures:

1. Because all *LEAVE*s branch to the same point, it is impossible, without retesting a flag or condition, to determine which exit was used.
2. Because *LEAVE* and *LOOP* or *+LOOP* continue after loop termination at the same place, it is impossible, without retesting a flag or condition, to determine how the loop was terminated.
3. There is no mechanism for directly exiting through multiple levels of *BEGIN* loops or *DO* loops. The only method currently available is to ripple a flag all the way out. Knowing how the loop was exited would also be useful here.

The last two problems apply only to common extensions:

4. Because all *WHILE*s in a *BEGIN* loop branch to the same

```
IF      THEN
IF      ELSE      THEN

BEGIN     UNTIL
BEGIN     WHILE     REPEAT

DO     LOOP
DO     +LOOP

DO     IF     LEAVE THEN     LOOP
DO     IF     LEAVE THEN     +LOOP
```

Table 1: *Forth 83 Standard control structures*

```
BEGIN     REPEAT
BEGIN     WHILE     WHILE ...  REPEAT
BEGIN     WHILE     WHILE ...  UNTIL

CASE     OF     ENDOF     OF     ENDOF ...  ENDCASE
```

Table 2: *Common nonstandard extensions*

Standard:
1. Can't handle each LEAVE exit separately without retesting for the exit condition after exit
2. Can't handle LOOP termination separately from LEAVE exits
3. Can't exit directly through multiple levels of begin-loops or do-loops

Common Extensions:
4. Can't handle each WHILE exit separately without retesting for the exit condition after exit
5. Can't handle UNTIL termination separately from WHILE exits

Table 3: *Limitations of standard control structures and common extensions*

Proposed Extensions

```
BEGIN   IF   LEAVES ... REPEAT   THEN
BEGIN   IF   LEAVES ... UNTIL   ELSE   THEN

BEGIN   IF   LEAVE THEN ... REPEAT
BEGIN   IF   LEAVE THEN ... UNTIL

BEGIN   BEGIN   IF   LEAVES ... REPEAT OUTSIDE   REPEAT   THEN
BEGIN   BEGIN   IF   LEAVES ... REPEAT OUTSIDE   UNTIL   ELSE   THEN

CASE   IF   LEAVES   IF   LEAVES ... ENDCASE

DO   IF   LEAVES ... LOOP   THEN
DO   IF   LEAVES ... +LOOP   THEN

DO   DO   IF   LEAVES ... LOOP OUTSIDE   (+)LOOP   THEN
DO   DO   IF   LEAVES ... +LOOP OUTSIDE   (+)LOOP   THEN
```

Suggested Additions for Efficiency

```
DO   ?LEAVE   LOOP
DO   ?LEAVE   +LOOP

DO   ?LEAVES   LOOP   THEN
DO   ?LEAVES   +LOOP   THEN

IF   IF ... IF   THENS
IF   IF ... IF   ELSES   THEN
```

***Table 4:** Proposed extensions and additions*

```
IF   THEN
IF   ELSE   THEN

BEGIN   UNTIL
BEGIN   REPEAT
BEGIN   WHILE   REPEAT
BEGIN   WHILE   WHILE ... REPEAT
BEGIN   WHILE   WHILE ... UNTIL

BEGIN   IF   LEAVES ... REPEAT   THEN
BEGIN   IF   LEAVES ... UNTIL   ELSE   THEN

BEGIN   IF   LEAVE THEN ... REPEAT
BEGIN   IF   LEAVE THEN ... UNTIL

BEGIN   BEGIN   IF   LEAVES ... REPEAT OUTSIDE   REPEAT   THEN
BEGIN   BEGIN   IF   LEAVES ... REPEAT OUTSIDE   UNTIL   ELSE   THEN

DO   LOOP
DO   +LOOP

DO   IF   LEAVE THEN   LOOP
DO   IF   LEAVE THEN   +LOOP

DO   IF   LEAVES ... LOOP   THEN
DO   IF   LEAVES ... +LOOP   THEN

DO   DO   IF   LEAVES ... LOOP OUTSIDE   (+)LOOP   THEN
DO   DO   IF   LEAVES ... +LOOP OUTSIDE   (+)LOOP   THEN

DO   ?LEAVE   LOOP
DO   ?LEAVE   +LOOP

DO   ?LEAVES   LOOP   THEN
DO   ?LEAVES   +LOOP   THEN

CASE   IF   LEAVES   IF   LEAVES ... ENDCASE

IF   IF ... IF   THENS
IF   IF ... IF   ELSES   THEN
```

***Table 5:** Proposed standard control structures*

(continued from page 31)

point (just after *REPEAT* or *UNTIL*), it is impossible to determine which exit was used without retesting a flag or condition.
5. When an *UNTIL* terminates a *BEGIN* loop containing one or more *WHILE*s, it is impossible to determine how the loop was terminated without retesting a flag or condition.

About 90 percent of the loops coded can be coded without too much effort. It is difficult, if not impossible, to code the other 10 percent without resorting to setting a variable or using some other unworthy method. Without solutions similar to those below, many programmers have spent hours trying to code reasonably efficient complex loops.

Proposed Extensions

Having collected this information, what can be deduced about the problem?

1. The exit/termination cases following loop execution need to be handled better.
2. The problems are identical for both *DO* loops and *BEGIN* loops.
3. The loop exit word must be executed conditionally to be useful.
4. The *THEN* following *LEAVE* is redundant because no code between *LEAVE* and *THEN* can ever be executed.
5. Given that the conditional exit test produces only *exit* and *don't-exit* results, there are only two cases that must be made programmable: the *exit* case and the *don't exit* case.

Given these deductions, the following preferences are likely:

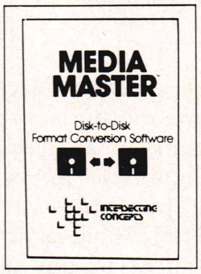
1. Because the problems are the same for both *DO* loops and *BEGIN* loops, a common syntax would be valuable—something such as *LEAVE* but more flexible.
2. Because the exit word must be used conditionally, introduction of the word with an *IF*-type word is necessary. A syntax similar to *LEAVE* is probable.
3. Because the *THEN* after *LEAVE* is redundant, our word should incorporate the *THEN* function.
4. Because the destination of our exit branch is determined by the programmer, a *THEN*-type destination marker is necessary.
5. Because the *don't-exit* condition continues execution after the *IF-ELSE-THEN* clause (at the old *THEN* point) and execution after *exit* resolves to execute elsewhere, our exit word could be considered analogous to *ELSE*.

We might now draw the following conclusion: A conditional exit with a structure similar to *IF-ELSE-THEN* would be appropriate, with the *IF-ELSE* part existing inside the loop and the *THEN* part indicating the destination. Furthermore, the current *IF-ELSE-THEN* words perform exactly the desired functions, except that the *ELSE*-type word needs to discard the loop parameters when exiting a *DO* loop. Hence, if a word were added to the *IF-ELSE-THEN* family to perform the appropriate *ELSE*-type func-

"MAKE YOUR IBM-PC CP/M COMPATIBILE"

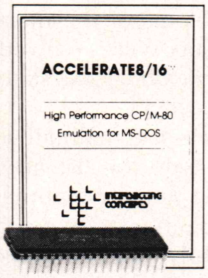
Intersecting Concepts Announces 3 Solutions To Solve Your Computer Incompatibility!

"But will it work on my computer?" *YES!* Finally, there are three *easy* ways to exchange information, transfer files, and run CP/M software on MS-DOS machines.



1. MEDIA MASTER™ is our direct disk-to-disk format conversion program. Already an accepted industry standard, this *$39.95** program uses simple screen prompts that lets you read, write and format up to 150 different 5 1/4" diskettes from CP/M, MS-DOS and PC-DOS operating systems. So if you work on a IBM PCompatible at the office, but use a CP/M computer at home, now you can easily transfer files that would otherwise be "foreign" to your computer's operating system.

2. MEDIA MASTER PLUS™ goes one step further by converting 8-bit CP/M software to run on 16-bit MS-DOS and PC-DOS machines. This newly released *$59.95* product combines our IBM-PC version of *Media Master* with *ZP/EM*, a powerful new emulation program. The results are amazing: CP/M programs using 8080 instructions *and* data can be transfered from popular computers like Osborne, Kaypro and Zenith to run on MS-DOS and PC-DOS machines!



3. ACCELERATE8/16™ is also new and dramatically improves the performance of *Media Master Plus* by tailoring the CP/M emulation around a NEC V20 microchip. This user installable plug-in chip simply replaces the 8088 processor in your MS-DOS computer. Once installed, it'll run your CP/M and MS-DOS software much faster. (Speed improvements are roughly 15% faster in MS-DOS and 350% faster in CP/M!) *Accelerate8/16* includes *Media Master Plus*, V20 CP/M Emulation Software, and the NEC chip for only *$99.95!*

All three solutions save you money by eliminating expensive modems and communications software!

TO ORDER

To order *Media Master, Media Master Plus,* or *Accelerate8/16*, call **800-628-2828, ext. 629.**

For additional product and upgrade information contact:

INTERSECTING CONCEPTS
4573 Heatherglen Court
Moorpark, CA 93021
or call *805-529-5073.*

Dealer inquiries invited. ** $99.95 for Dec Rainbow*

Circle **no. 296** on reader service card.

(continued from page 33)

tion, a complete structure and syntax would exist.

The proposed word is *LEAVES* and belongs to the *IF-ELSE-THEN* family. The name is appropriate, being similar to *LEAVE*. It reads well, and it expresses immediacy. The word *OUTSIDE* is also proposed to allow exiting several levels of *BEGIN* loops or *DO* loops directly. Examples of their uses are represented in Table 4, page 32. The additional words presented in this article are proposed to be part of the standard as a new word set, or even better as a new level above the existing Required Word Set.

Implementation

As described above, the Forth 83 *LEAVE* was not obvious to implement. It does, however, open the door to the control structure implementation solution. Thanks to David Harralson, whom I met at the 1985 FORML Conference in Asilomar, California, for extending the concept. He presented a paper there [2] that seemed to solve all the problems (and more) but completely lacked compatibility with the current standard. We formed a working group at the meeting to discuss the problem, spent several hours on the telephone, and exchanged several letters during the months that followed. His functional but nonstandard implementation, combined with a partial solution I had already achieved,[3] evolved into the results seen here. By working together we were able to extract the key concept from his paper: All forward references are to be maintained in linked lists during compilation.

We actually took his paper and worked backward, eliminating the overgenerality of his structures that prevented them from being standard compatible. I then resolved the syntax difficulties and decided upon behavioral rules for *LEAVES* and *OUTSIDE*. Finally, I spent many hours staring at the walls and scrawling on paper until one by one the implementation problems of the more conservative standard-compatible syntax were resolved. Out of this also came an almost free, flexible *CASE* statement. Listing Three, page 82, is the result.

As mentioned, all forward references are maintained in linked lists. Harralson prefers to keep the list head pointers on the stack; I prefer to use variables (sometimes a useful poison). I find the variable implementation much more clear even though it may be slightly larger. It is also more easily modifiable if additional forward referencing structures are added in the future.[4]

Three words need to be added to the System Extension Word Set to handle forward compiled linked lists: *>MARKLIST*, *>RESOLVELIST*, and *>RESOLVESLIST*. (See Listing Three.) Notice that the first and last words are already in the system if approach 2 (Listing One) is taken to implement the standard *LEAVE* operation. If the extended control structures presented here are adopted, I would expect that all three words have a very good chance of becoming standard.

Three lists are maintained by the system:

1. The *IF-LIST* links all *IF* and *ELSE* branches. These branches are resolved by *ELSE* and *THEN* respectively. The *IF-LIST* is also used to hold the unresolved *LEAVES* branches once outside the loop structure. This is how *LEAVES* can be resolved by *ELSE* or *THEN*.
2. The *LEAVES-LIST* contains all *LEAVES* branch addresses while inside a loop. The list is transferred to the *IF-LIST* after the loop end is compiled to allow the *LEAVES* to be resolved by appropriate *ELSE*s or *THEN*s.
3. The *LEAVE-LIST* is a list of *LEAVE* branch addresses, maintained for compatibility with the current *LEAVE* function. The list is resolved by *LOOP*, *+LOOP*, *REPEAT*, and *UNTIL*. *LEAVE* can be used inside either *DO* loops or *BEGIN* loops.

Another variable maintained by the system is *LEAVE-CF*. This variable is the key that allows *LEAVE* and *LEAVES* to work in both *BEGIN* loops and *DO* loops. *CASE*, which is used by *BEGIN*, and *DO* set the value of *LEAVE-CF* to *BRANCH* and *LEAVE* respectively. *LEAVE-CF* thus contains the compilation address (code field) of the proper run-time routine to be compiled by *LEAVE* or *LEAVES*, depending upon whether a *BEGIN* loop or a *DO* loop is being compiled. The compilation addresses of other exit routines can, of course, be stored in *LEAVE-CF*. This allows *LEAVE* or *LEAVES* to be used to exit almost any structure added to Forth. To allow proper structure nesting, all list heads (*IF-LIST*, *LEAVES-LIST*, and *LEAVE-LIST*) and the value of *LEAVE-CF* must be saved during compilation and the lists set to 0 at the beginning of each new structure (*BEGIN*, *DO*, and *CASE*). The heads' values and *LEAVE-CF* are restored at the end of each structure (*UNTIL*, *REPEAT*, *LOOP*, *+LOOP*, and *ENDCASE*).

Comparing the complexity of the new implementation of the *IF-ELSE-THEN* structure in Listing Three to that in Listing One, you can see that there is little significant difference. Also, comparing the complexity of the new *BEGIN-WHILE-REPEAT* to that in Listing One (ignoring *LOOPEND* for a moment) or better yet to the commonly extended version in Listing Two, you can see that here too there is little significant difference. The proposed structures' complexity is even less in one area, as *WHILE* is now an alias for *IF*. The *DO* loop words also follow this tradition with no significant difference in complexity (*LOOPEND* somewhat aside).

Capability is not free, and complexity rears its ugly head in *LOOPEND*. First, within *LOOPEND*, the backward branch is resolved to the beginning of the loop, and then an entire linked list of branches is resolved to the point immediately following the end of the loop. For *DO* loops this is the *LEAVE-LIST*, and for *BEGIN* loops it is the *IF-LIST* (to resolve the *WHILE* exits). *BEGIN* loops also resolve the *LEAVE-LIST* after *LOOPEND*. If any *LEAVES* were compiled, the list heads are restored and then the end of the *LEAVES-LIST* is found and is linked to the front of the *IF-LIST*. The new longer list replaces the *IF-LIST*. This slightly complex but necessary process allows *THEN* to be used to resolve *LEAVES*. The process might be simplified if two additional words were added to replace *ELSE* and *THEN* in resolving operations on the *LEAVES-LIST*. This unfortunately interferes with the operation of *OUTSIDE*. Harralson and I feel that these are simply two more words added to remove a small amount of complexity that can be hidden within the bowels of the system word *LOOPEND*, especially when the *IF-LEAVE-THEN* syntax must be retained for compatibility with the current standard.

NOW YOU CAN TAKE IT WITH YOU WHEN YOU GO

TECH PC Portable computers are designed for travel. Each unit is housed in an attractive yet functional case designed to allow maximal expandability with minimal difficulty. The units can be configured to be as powerful as any desktop and yet have the mobility of a fully portable unit. These units are truly designed to meet the needs of the individual on the go.

The TECH PC/XT PORTABLE weighs in at less than 40 pounds and is ideal for the individual who has limited desk space or the business executive who is often traveling.

Like our TECH PC DESKTOPS, the TECH PC PORTABLES run all software designed for the IBM PC, and compatibles, such as: dBase III, Lotus 123, Symphony, Wordstar, Wordperfect, Framework, programs based on MS-DOS, PC-DOS, PICK, CP/M86, and UCSD p-system operating systems.

TECH PC PORTABLE COMPUTERS are available now in 4 different base models:

TECH PC/XT PORTABLE .... $ 999
Options:
Tech Turbo PC/XT Portable with 20 MB Hard Disk $1499
Tech Turbo PC/XT Portable with 1200 Baud Internal Hayes Compatible Modem and 20 MB Hard Disk $1699

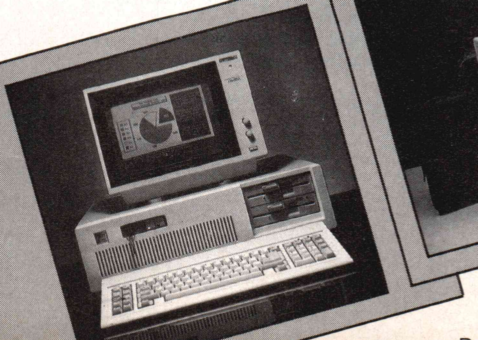
TECH TURBO PC/XT PORTABLE ... $1095
Options:
Tech Turbo PC/XT Portable with 20 MB Hard Disk $1595
Tech Turbo PC/XT Portable with 1200 Baud Internal Hayes Compatible Modem and 20 MB Hard Disk $1795

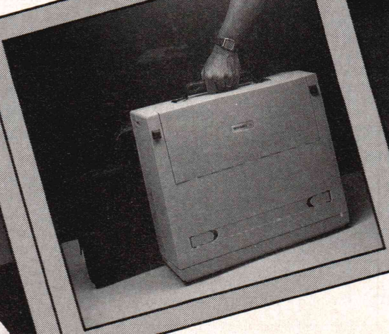
TECH PC/AT PORTABLE $1999
Options:
Tech PC/XT Portable with 20 MB Hard Disk $2299
Tech PC/XT Portable with 1200 Baud Internal Hayes Compatible Modem and 20 MB Hard Disk $2499

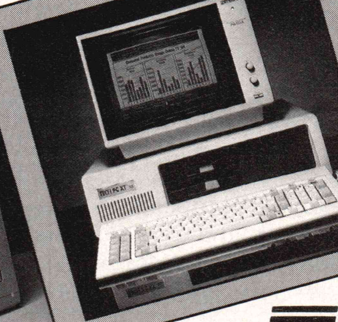
TECH TURBO PC/AT PORTABLE ... $2099
Options:
Tech PC/XT Portable with 20 MB Hard Disk $2399
Tech PC/XT Portable with 1200 Baud Internal Hayes Compatible Modem and 20 MB Hard Disk $2599

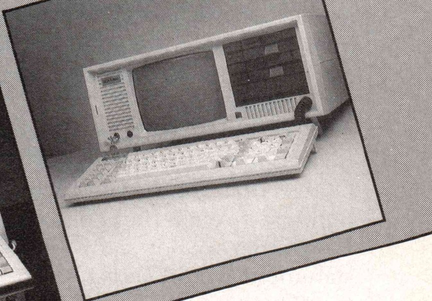
All TECH PC PORTABLES available with tape backups, hard disks up to 280 megabytes, networking systems, and hundreds of other hardware and software accessories.

TECH PERSONAL COMPUTERS is a full service manufacturer of Micro Computer Products and offers a complete line of Desktop, Portables and Multi-User Computer Systems as well as an accessory line of over one hundred enhancement products. **TECH PERSONAL COMPUTERS** are all backed by a full one year warranty with additional maintenance coverage and extended maintenance contracts available through Mohawk Data Sciences. For more information concerning hundreds of MDS Service Centers throughout the United States, contact **TECH PERSONAL COMPUTERS** at **(714) 754-1170.**









TECH PC

714/754-1170

2131 S. Hathaway, Santa Ana, California 92705

TELEX: 272006 Answer Back-TECH FAX: 714/556-8325

Dealers only circle no. 245 on reader service card. Users only circle no. 279 on reader service card.

(continued from page 34)

The last looping-related word is *OUTSIDE*. This word must keep the branch address of the most recent *LEAVES* available to be resolved outside the next level of loop. This is accomplished by simply unlinking the top item from the *IF-LIST* into the the top of the *LEAVES-LIST*. Outside the next level of loop, the branch address will be transferred back to the *IF-LIST* again by *LOOPEND*. This process can be repeated as necessary. When multiple *LEAVES* are used, *OUTSIDE* can be placed between *LEAVES* resolutions to work upon the correct one. *OUTSIDE* must also ensure that *LEAVES* unnests the correct number of *DO*-loop levels. Unfortunately, I cannot disclose this mechanism because it is proprietary to my company's product. Note that because of this difference in unnesting requirements, the listed code for *OUTSIDE* will not properly unnest through a nested mixture of loops. Though easy to solve, this too is proprietary.

The proposed structures give you an almost free implementation of a *CASE* statement. It compiles in a manner exactly equivalent to nested *IF-ELSE* statements but with a more readable syntax and a single syntactic nesting level. This is equivalent to the *else if* clauses available in C, Pascal, Ratfor, and Ada.[5] It can also be implemented more directly by implementing *THENS*. (See Listing Four, page 83.) Similarly, the *ANDIF* proposal can be easily implemented by defining *ELSES*.[6] (See the examples in Listing Five, page 83.)

The new *LEAVES* structure seems even more unstructured than the old *LEAVE*. This is not so. The single exit point of the module simply needs to be redefined. The program flow rejoins at the *THEN* of the outsidemost *LEAVES*—hence one entry point, one exit point. In another puff of logic, even *LEAVES* is somewhat structured. Working with the same rules, when *OUTSIDE* is considered, the module becomes the outsidemost *DO* to the outsidemost *THEN*, which resolves a *LEAVES* within. It is a much harder logical stretch, but even when *OUTSIDE* is considered, this oft-broken rule of structured programming is somewhat fulfilled.

Conclusions

Listing Five lists examples of almost every control structure proposal published or presented to date and the corresponding solution using the structures proposed in this article. All the bases seem to be covered. Some proposals may have been omitted, but hopefully none represent structures that are not adequately covered by the included proposals.

Using linked lists for all forward references might be questioned. Creating a new *IF*-type structure for the *LEAVES* operation would have solved that class of problem just as well, though it would have added additional words that are not really necessary. The almost-free *CASE* statement would vanish in an imposed limitation of syntax. This would complicate *OUTSIDE* and preclude the useful *ELSES* and *THENS* operations without adding words to mark their limits. (See the *<STEPS* example in Listing Five.)

Table 5, page 32, summarizes the syntax of the control structure word set proposed for standardization. It is completely compatible with the current standard control structures and the most common nonstandard extensions. It appears to emulate all the tricks that have been proposed to date for these classes of control structures. It implements in 19 words (28 including the System Extension Word Set) all the described functions compared to 33 words (41 including the System Extension Word Set) to summarize the other proposals. The current standard has 11 words (17 including the System Extension Word Set). The material in this article was presented to the March 1986 meeting of the San Francisco Chapter of the Forth Interest Group, and the members voted two to one in favor of standardization. I hope the enthusiasm continues.

Notes

1. David W. Harralson, "Extending FORTH Control Structures into the Language Requirements of the 1990's," *Seventh FORML Conference* (1985).
2. Ibid.
3. George W. Shaw, "Extended Control Structures for Forth," *Seventh FORML Conference* (1985).
4. Harralson, "Extending FORTH."
5. Wil Badin, "Modern Control Logic," *Fifth FORML Conference Proceedings* (1983).
6. Wendell C. Gates, "ANDIF and ANDWHILE," *Forth Dimensions*, vol. VI, no. 4.

Bibliography

Gray, R. W. "DO . . WHEN . . LOOP Construct." *Forth Dimensions*, vol. VI, no. 6.

Harralson, David W. "FORTH Control Structures." *Forth Dimensions*, vol. VI, no. 2.

Harris, Kim R. "Proposed Extensions to Standard Loop Structures." *Fifth FORML Conference Proceedings* (1983).

Harris, Kim R. "Transportable Control Structures." *Rochester Forth Standards Conference* (1981).

Hore, Michael. "Enhanced DO LOOP." *Forth Dimensions*, vol. VI, no. 6.

Komusin, Bruce. "A Generalized Loop Construct for FORTH." *Forth Dimensions*, vol. 2, no. 2.

Main, Richard B. "FORTH-85 'CASE' Statement." *Forth Dimensions*, vol. 5, no. 6.

Sanderson, Dean. *Forth Standards Team Proposal*, no. 283 (1983).

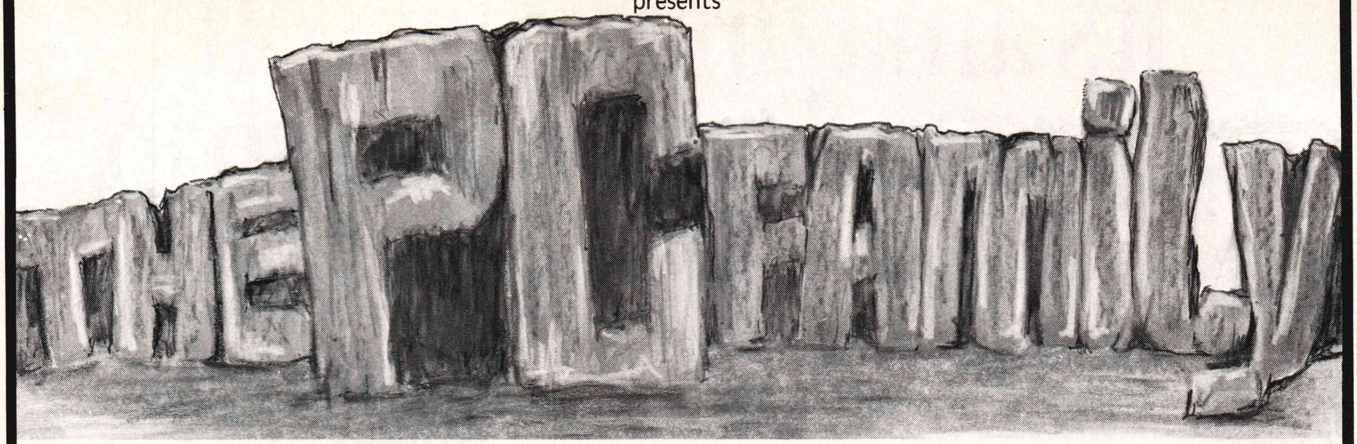
DDJ

(Listings begin on page 82.)

Indicate your opinion of standards proposed in this article.
For standardization: Circle Reader Service **No. 3.**
Needs work before standardization: Circle Reader Service **No. 4.**
Against standardization: Circle Reader Service **No. 5.**

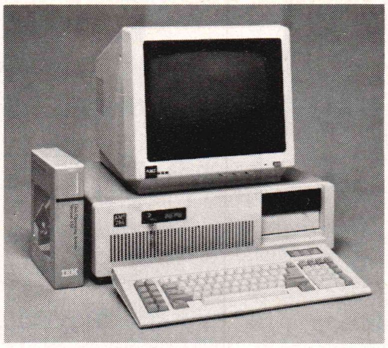
Vote for your favorite feature/article. Circle Reader Service **No. 6.**

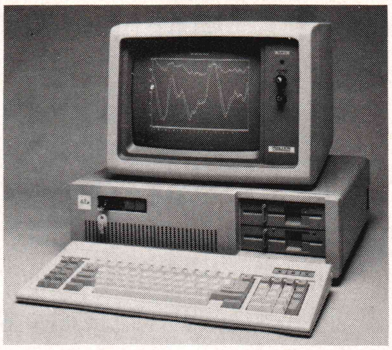
AMERICAN MICRO TECHNOLOGY
presents



STARRING

XT-PLUS AMT 286 and Introducing AT*jr*





XT-PLUS

IBM PC XT Compatible, 4.77MHz Clock, 640K Mother Board, 8088 Intel Chip, Keyboard, 135 watt Power Supply, Floppy Disk Controller, Printer Port, Serial Port, Game Port, Clock w/Battery Backup, Two Disk Drives ...... $699.

Specials *

Floppy Disc Controller ...... $29.
Monochrome Graphics Card/PP .... 79.
Disk I/O Card FDC, PP/SP, Game, Battery Clock ...... 79.
1200 Baud Modem (½ size) .. 159
XT-Mother Board "O"K expandable to 640K .... 99.
Mother Board for AT ...... 650.
Keyboard AT .... 65.
20MB Drive with Controller .. 425.
Power Supply 135 watts .... 65.
Power Supply 200 watts ... 129.
XT Chassis .... 31.
AT Chassis .... 89.
8MHz IBM PC XT Compatible Mother Board "O"K ...... 175.
30MB Drive with Controller .. 590.
EGA Card .... 275.

AMT 286

IBM PC AT Compatible Computer (STM Board) 6 MHz, 640K Memory, Keyboard, Clock & Battery on Board, Floppy & Hard Disk Controller, 1.2MB Floppy, 192 watts Power Supply .... $1499.*

AMT 286-e

IBM PC AT Compatible Computer (Atronics Mother Board), 8 MHz Clock (enhanced version), 640KB Memory expandable to 1MB, AT layout Keyboard, Floppy & Hard Disk Controller by Western Digital, 192 watts Power Supply, 1.2MB Floppy, 20MB Hard Disk, Socket for 80287 .... $1999.

"O" Wait State Option .... $300.

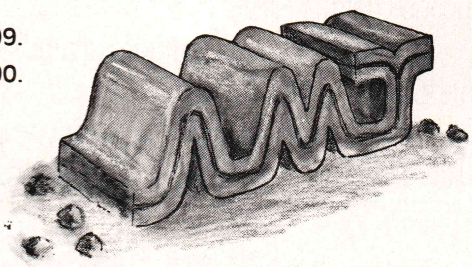
PLEASE DON'T CALL TICKETRON . . .
FOR FAST SERVICE CALL US

(714) 972-2945
TWX 5106003265
(213) 477-6320 inside Los Angeles

AT*jr*

IBM PC Compatible, 2 to 5 times faster than IBM PC, Zero Wait State, Eight Slots, 135 watts Power Supply, 8 MHz & 4.77 MHz Clock (Dual Speed Hardware & Software switchable), 640KB on Mother Board, V-20 or 8088-2 Processor, AT layout Keyboard, Floppy Disk Controller, Two Drives 360KB each, Runs Lotus 123, Wordstar, dBase II & III, Flight Simulator and more .... $699.*

*Prices for quantity purchases only
Registered trademark of IBM Corporation



AMERICAN MICRO TECHNOLOGY
1322 E. EDINGER SANTA ANA, CA 92705

Prices and Availability subject to change without notice.

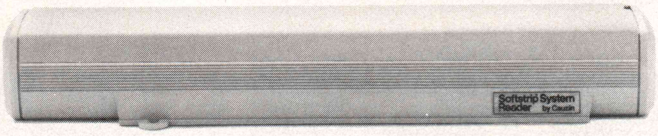
Circle **no. 297** on reader service card.

It's amazing what you can reveal when you strip.

Introducing a shape that's about to turn on an entire industry.

The Softstrip™ data strip. From Cauzin.

This new technology allows text, graphics, and data to be encoded on a strip of paper, then easily entered into your computer using a scanning device called the Cauzin Softstrip™ System Reader.



The Cauzin Softstrip System Reader replaces tedious typing by scanning the strip and reading it into your computer.

Creating a simple, reliable and cost efficient way to distribute and retrieve information.

Softstrip data strips, like those you see here, can contain anything that can be put on magnetic disks.

Facts. Figures. Software programs.

Video games. Product demonstrations. Sheet music.

The Cauzin Softstrip System Reader is now compatible with the IBM PC, Apple II and Macintosh.

A single strip can hold up to 5500 bytes of encoded data.

It can stand up to wrinkles, scratches, ink marks, even coffee stains.

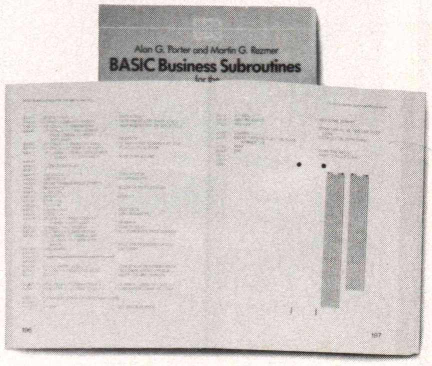
And it can be entered into your computer with a higher degree of reliability than most magnetic media.

Simply by plugging the Cauzin Reader into your serial or cassette port and placing it over the strip.

The reader scans the strip, converts it to computer code, and feeds it into any standard communication interface.

Because strips are so easy to generate, most of your favorite magazines and books will soon be using them in addition to long lists of program code.

And you'll be able to enter programs without typing a single line.



Soon everyone will be stripping as data strips appear in popular magazines, computer books and text books.

There is also software for you to generate your own strips. Letting you send everything from correspondence to business information using our new technology.

Find out how much you can reveal by stripping. Just take this ad to your computer dealer for a demonstration of the Cauzin Softstrip System Reader.

Or for more information and the name of the dealer nearest you, call Cauzin at 1-800-533-7323. In Connecticut, call 573-0150.

Softstrip™
COMPUTER READABLE PRINT

Cauzin Systems, Inc.
835 South Main St., Waterbury, CT 06706

Apple® and Macintosh® are registered trademarks of Apple Computer Inc., Apple® is a registered trademark of Apple Records, Inc., Softstrip® and the Softstrip® System Reader are trademarks of Cauzin Systems, Inc., IBM® is a registered trademark of IBM, Inc.

Circle **no. 226** on reader service card.

MAKING THE GOOD LIFE EVEN BETTER

Someone once said that there is nothing new under the sun. Wouldn't life be boring if that were indeed true? The data strips on the right contain the program described in the article "The Game of Life in Expert-2", by Jack Park, which appears in this issue. It's a prime example of how something, in this case the game of **LIFE** itself, can, indeed be improved.

The game of **LIFE** was invented years ago by John Horton Conway. Over the years, the game has evolved into a popular cerebral exercise for programmers and math majors alike. At first the game was played on graph paper, but the advent of modern technology moved it to the computer which plays the game thousands of times faster. Now millions of computer enthusiasts are captivated by this devilishly simple, yet marvelously complex quintessential computer diversion.

The rules of the game are quite simple. Imagine that you have an infinite grid of squares, each one being either alive (on) or dead (off). Each square (called a "cell") lives or dies into the next cycle (called a "generation") based on its current state and that of its neighbors. The grid of cells is represented by a graphic display on your computer screen. After setting up an initial configuration of living and dead cells, you start the simulation. The patterns will change on the screen as cells live and die.

Mr. Park's improvement on the theme is interesting because of his approach. Instead of writing a traditional program for the simulation, he has created an array of intelligent cells using an inference engine written in Expert-2, a superset of FORTH.

Read in the data strips, following the directions that came with your Cauzin reader. You'll need the Expert-2 programming environment to operate this program. Refer to Mr. Park's article in this issue for operating instructions.

Reprinted with permission of Dr. Dobb's Journal.

StripWare Library No. 202

1 2 3

Softstrip
COMPUTER READABLE PRINT

Forth Goes to Sea

by Everett Carter

Forth has a long history in process control and data acquisition, but its use in oceanography is new.

This article describes an implementation of the Forth language for the control of a microprocessor-driven laboratory instrument. The laboratory in this case is the ocean; the processor is the Motorola MC146805, a CMOS 6805 chip. I implemented a ROM-based Forth for this chip following the Forth 79 Standard as closely as possible, but the final version (Listings One and Two, pages 84 and 88) has several differences from the standard. The deviations from the 79 Standard are because there is no mass storage and all the code must reside in ROM. Memory constraints are also important because everything—the managing software and the data itself—must fit within 8K.

The RAFOS Float System

Our research group in the Graduate School of Oceanography at the University of Rhode Island has spent the last two years studying the Gulf Stream current in the Atlantic Ocean. The primary instrument we have been using is known as a RAFOS float.

The origin of the name is somewhat convoluted. In the ocean there is a sound conducting channel, like a waveguide, called the SOFAR channel. Its depth varies, but in the Atlantic it's about 800 meters deep. Early in the 1970s, this channel was utilized in a neutrally bouyant float. These early floats drifted in the channel emitting sound pulses at precisely known times. Listening stations on shore picked up the signals, and by triangulating from several stations, the float positions were determined. These floats were called SOFAR floats. We're now using new floats that listen passively to fixed sound sources. They thus work in the opposite fashion from SOFAR floats, hence their name RAFOS, which is SOFAR spelled backward.

Everett Carter, Graduate School of Oceanography, University of Rhode Island, Kingston, RI 02881

The RAFOS float is made of Pyrex glass, is about five feet long and about four inches in diameter, and looks like a big test tube. Inside it are various sensors, batteries for power, a radio, and an MC146805 microprocessor that controls them all.

Our sampling scheme involves tossing the float into the water (whereupon it sinks to the proper depth) and letting it drift freely for 30 to 45 days in the Gulf Stream. After the allotted time, the float drops its ballast weight and comes to the surface. It then turns on the radio and transmits its data to shore via satellite. We make no attempt to recover the float—it would cost more in ship time than it cost us to build the float.

Beginning last fall, we have been contemplating how to utilize the next generation of floats. These floats will be used as before but with adaptive sampling strategies or for shorter intervals interactively with a ship. An interactive scenario could involve, for example, putting a few floats in for a few days and recovering them from the same ship, which then puts in more floats in a way that depends upon the data received from the first floats.

The original RAFOS floats were all programmed in assembly language, and all the floats had the same program. Assembly code for the interactive floats would be a nightmare (imagine writing machine code late at night, while seasick, so you can do the experiment the next day!). Forth was the obvious solution to the problem of how to get the sampling program into the float efficiently.

The MC146805 Processor

As mentioned previously, the processor in the float is the Motorola MC146805, which is basically a CMOS 6805. The 6805 is a 6800 that has been specialized for process controlling. It was the only CMOS alternative to the 1802 when the floats were first designed several years ago. It is not the ideal choice for the implementation of Forth. In specializing from the 6800 to the 6805, half the registers and several instructions were thrown away and the stack could not be accessed (it saves only return addresses for calls and interrupts and the registers during interrupts).

One class of instructions that are particularly useful in implementing Forth are the indirect ones. An indirect jump instruction causes the processor to jump to an address that was pointed to by the contents of a location, and that location is identified by a register or other memory location. Some processors have a whole family of indirect instructions, jumps, reads, writes, and so on. Some of these instructions I learned to live without; others I had to emulate in self-modifying software.

The 6805 comes in several versions. The one I am using has a memory address space of only 8K, which puts severe restrictions upon the system because everything—the operating system, the application program, the memory-mapped sensor ports, and the data—must all coexist in only 8K. In the early floats that were programmed in assembly language, the code occupied the top 4K (from 1000

hex to 1FFF), and the data was stored in the lowest 4K. This Forth implementation tries as much as possible to preserve the historical partition (it nearly succeeds).

The Format of the Header

In order to conserve the system's memory usage, the structure of the headers is modified slightly. Only the character count and the first three characters are saved in the header. This is as if the Forth variable *WIDTH* was set to 3 (note that *WIDTH* does not explicitly exist in this implementation and that all words that would use it treat it as a constant that is equal to 3). The count byte has the following structure. The natural character count is in the low 5 bits (hence allowing words up to 31 characters long). The sixth bit is the smudge bit; it equals 1 when the word is smudged, thus preventing a match by *−FIND* and similar words. The seventh bit has no defined use. The eighth bit is normally 0 but is set to 1 for immediate words (this bit is called the precedence bit). Although strictly this structure does not violate the standard, it is not the usual one. Traditionally, bit 8 is always set to 1 and bit 7 is the precedence bit. The traditional structure was not used in order to make *WORD* smaller—*WORD* masks out the high-order bit of every byte that it examines in a given word's name field when it is trying to find a match. In the traditional format, the count byte has a different mask from that of the rest of the name field.

The format of the header is thus the count byte, followed by the first three characters of the word, a pointer to the name field (that is, the count byte) of the previous Forth word, and finally the actual code for the word. Note that last point. After the link is the actual code—that is, the code field, not the code field address.

The Inner Interpreter

The Forth interpreter is implemented as a direct threaded interpreter because of the limited ability of the 6805 processor to perform address indirection. My reading of the standard is that it does not specify the type of interpreter you should use, but the interpreter is usually implemented as an indirect threaded interpreter. The direct threaded approach means that the code field is not pointed to by a pointer, but the code actually begins where a traditional *CFA* pointer would normally be (so that ' returns the address of the code field, not the address of the pointer to the code field). This means that you must exercise care in using Forth words that manipulate addresses (*CFA*, *PFA*, *NFA*, *'*, and so on) if you are accustomed to using more traditional Forth implementations.

Forth in ROM

Even though this implementation of Forth is designed to be ROM-based, some code resides in RAM for special reasons. The 6805 has special, fast instructions for access to the base page (0000 to 00FF hex). The instructions are fast because fewer bytes are used in those instructions and because some of the base page resides physically within the processor chip itself. Because the processor will spend the bulk of its time in the inner interpreter, the inner interpreter is designed to reside in the base page (starting at 0080 hex).

All the self-modifying code must reside in RAM in order to work; this code immediately follows the inner interpreter in RAM. The limited instruction set of the 6805 required the simulation of some instructions (such as indirect jumps) through the use of self-modifying code. In many places I found I needed two routines—one to load register *A* into a pointed-to address (an indirect write) and one to

TYPE	EXIT	EXECUTE	EMIT	BL
WORD	<NUMBER>	DROP	C@	@
DP	HERE	NOT	1+	HLD
STATE	CONTEXT	CURRENT	FORTH	!
C!	,	C,	DUP	+!
LATEST	ALLOT	LIT	COLD	QUIT
SWAP	SP!	CR	CREATE	TOGGLE
IMMEDIATE	−FIND	COUNT	0	1
2	2+	[	]	DEFINITIONS
+	−	U*	U/MOD	S−>D
PAD	<#	OVER	#>	>R
R>	R@	ROT	HOLD	M/MOD
BASE	SMUDGE	ABS	0<	0=
<	>	=	SIGN	NEGATE
+−	#	OR	AND	XOR
DDUP	#S	.	COMPILE	;
:	'	VARIABLE	CONSTANT	*
[COMPILE]	BEGIN	AGAIN	UNTIL	IF
THEN	ELSE	WHILE	REPEAT	<.">
TIB	>IN	'STREAM	<DO>	<LOOP>
<+LOOP>	DO	LOOP	+LOOP	DNEGATE
I				

Table 1: *List of initial Forth words*

DP	01D0	the dictionary pointer
START	1E57	where the outer interpreter is
BASE	10	initial base is hex (note that this is a single-byte variable)
FORTH	17E6	points to the NFA of the last dictionary entry (I)
CONTEXT	0039	points to FORTH
CURRENT	0039	points to FORTH

Table 2: *The initial values of the system variables. All values are hexadecimal.*

(continued from page 41)

get the value at that address into *A* (an indirect read). I called these routines *LOAD* and *GET* in the assembly listing. For generality, I wrote them to include a possible offset defined by register *X*. Many routines will define the 16-bit address defined at *LOAD+1* or *GET+1* and call *LOAD* or *GET*. Because they get modified, *LOAD* and *GET* are in RAM, and because they can be called frequently, they are in the base page.

The self-modifying code goes beyond the base page, but I made an effort to place the most frequently called portions within the base page. The predefined user variables (*FENCE, STATE, FORTH, CONTEXT, CURRENT, BASE, HLD, DP, IN,* and *OUT*) and the system variables (*IP, RP,* and *SP*) also reside in the base page.

When the system is booted, reset, or upon execution of *COLD*, the initial values of the user and system variables, the inner interpreter, and the self-modifying code are all copied from ROM to their executable locations in RAM. A note about how the code was developed: The assembly code was cross assembled on a VAX 750 using the XASM6805 cross assembler from Intelligent Devices of Minnesota (P.O. Box 492, Anoka, MN 55303). The IDM cross assembler does not allow the assembly of code at one location for execution at another. The code was thus written for its target address, then the hexadecimal assembly output file (which is in Motorola's S1 format, similar to the CP/M HEX format) was edited manually to change the compilation address for the dozen or so lines that needed changing.

The initial vocabulary consists of the words listed in Table 1, page 41; the system variables are initially set to the values shown in Table 2, page 41. The vocabulary is not a full Forth word set, but most of the important (non-mass-storage) words are there. We usually upload all the MOD arithmetic words from a microcomputer that we use as a terminal when we use the float; this is how the data-sampling words get loaded as well. The initial system memory map is shown in Figure 1, left. The variable *START* (at address 002E hex) does not have a header. It is a warm boot execution vector (that is, the address at that location is executed when *QUIT* is invoked), and normally it points to the start of the outer interpreter. A word that is pointed to by *START* should be a Forth word that is an infinite loop (such as a *BEGIN... AGAIN* structure) because it is effectively the outer interpreter once *QUIT* is executed. Note that *QUIT* is a warm boot; it does a partial system reset. It clears the stacks and sets the system state to execute, then it goes to the address pointed to by *START*. *COLD* runs the same code as the power-up or reset interrupt does—it resets the dictionary pointer and *FORTH* to their power-up default values and sets the *CONTEXT* and *CURRENT* vocabularies to *FORTH*. (All this is done by executing the ROM-to-RAM copy described above.) It then falls through to *QUIT*.

Error Checking

In order to minimize the size of the system, only a minimal amount of error checking is implemented. If the system does not find a word in the dictionary and it cannot subsequently interpret it as a number, that word is echoed back to the terminal, followed by a question mark. When this happens the user stack is unaffected, but the *FORTH* return stack is cleared. Clearing the return stack effectively denests the system from any process out to the outer interpreter. The system is then waiting for user input. If the system was in the compiling state when the error occurred, it will be in the execution state after the error message. This means that if a word was not found during compilation,

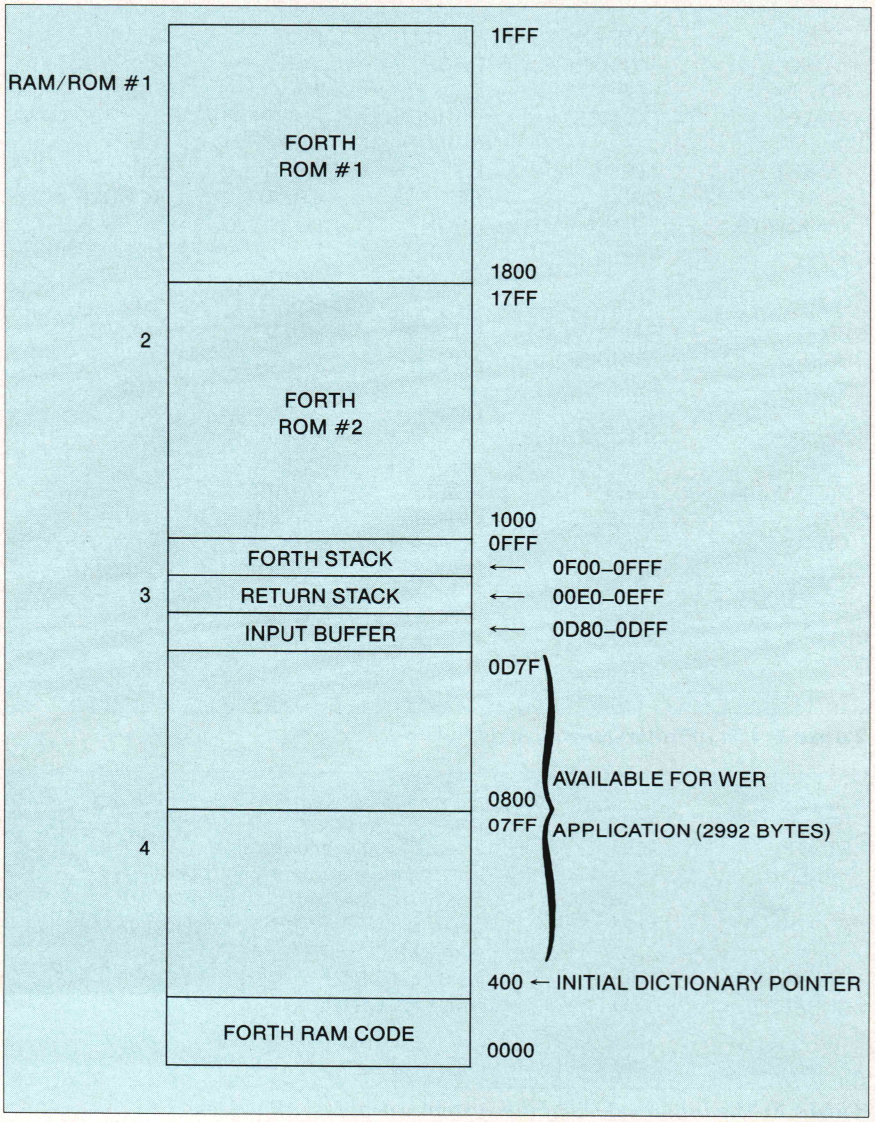


Figure 1: Memory map

Put RUN/CPM™ in your PC today and run 50,000 programs that you couldn't run yesterday!

Here is what the reviewers are saying about "RUN/CPM."

"It's an excellent program backed by a knowledgeable company."
ROBERT ATHEY
Micro Times

". . . the most powerful of all the emulators reviewed."
TED DRUDE
Computer Shopper

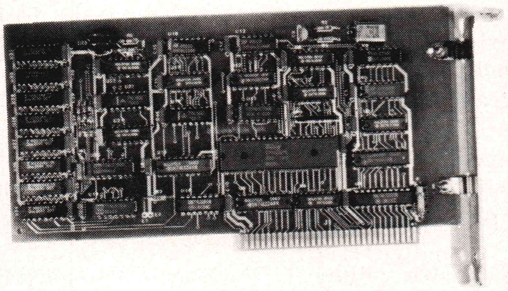
RUN/CPM V-20/30 CP/M Emulator

RUN/CPM is a revolutionary new product which combines software and hardware to allow your PC to run 1,000's of programs written for the CP/M80 operating system. For years, popular 8 bit computers such as "Kaypro", "Osborne", and "Morrow" have relied on this vast library of high quality, inexpensive software, which can now be accessed by your PC. A simple replacement of your PC's 8088 or 8086 microprocessor with a dual 8/16 bit N.E.C. V-20 or V-30 microprocessor creates a system capable of running both CP/M80, and MS-DOS/PC-DOS software. In addition to adding 8 bit CP/M compatibility, your PC's overall performance will improve by up to 30% as a result of the speedup characteristics of the V-20 and V-30 microprocessors. Running CP/M software directly from CP/M disks is easy thanks to RUN/CPM's disk emulation feature, which will allow your PC's floppy drives to directly READ, WRITE, and FORMAT the most popular CP/M disks. You will also be able to run CP/M programs 2 to 3 times faster, in full color, from a hard disk or RAM disk, and actually run MS-DOS background programs such as Borland's "Sidekick" on top of CP/M programs! Add pop-up windows, terminal emulation, and help screens and it is easy to see why RUN/CPM is fast becoming the defacto standard in CP/M emulation software.

READ/CPM CP/M Disk Emulation

READ/CPM adds CP/M disk compatibility to your PC, XT, or AT. Now your PC's floppy drives can directly access data disks from approx. 100 of the most popular CP/M formats using DOS commands "COPY", "TYPE", "DIR" etc.. This product does not require any additional hardware.

Run CP/M® Z80 for the incredible price of $199.95



The RUN/CPM Z-80 co-processor gives you all of the incredible features of RUN/CPM, plus the advantage of the full Z-80 instruction set at a price you can afford. Never before has it been easer to add Z-80 compatibility to your PC, XT, or AT. The RUN/CPM Z80 co-processor board comes complete with 64K of memory, a 5Mhz clock, and of course the incredible RUN/CPM emulation software. Plug this half size card into one of your PC's slots and within minutes you will be running the most powerful CP/M software available, including the most popular assemblers, compilers, debuggers, and wide variety of application programs such as word processors and spreadsheets. Our ISIS-II emulation option will even allow you to run "Intel" development languages on your PC, or AT. If you want to step up to the ultimate in high speed Z80 processing, then check out our RUN/CPM Z80-H full size co-processor available for only $499.95. Running in excess of 9mhz with 128 K memory on board, this co-processor virtually blows away the competition! Both the RUN/CPM Z80 & Z80-H co-processors come with the RUN/CPM emulator software and are backed by our 30 day money back guarantee.

I WANT TO ORDER!

To order by phone,
or for a dealer nearest you,
CALL (800) 637-7226
Technical Info call (305) 823-8088

Copies	Product	Price	Total
____	RUN/CPM requires V-20/30	$ 99.95	____
____	RUN/CPM Z80 (Board 5mhz)	$199.95	____
____	RUN/CPM Z80H (Board 9mhz)	$499.95	____
____	READ/CPM	$ 49.95	____
____	V-20 8mhz - Replaces 8088 up to 8 Mhz	$ 24.95	____
____	V-20 5mhz - Replaces 8088 up to 5 Mhz	$ 14.95	____
____	V-30 8mhz - Replaces 8086 up to 8 Mhz	$ 29.95	____
____	V-30 10mhz-Replaces 8086 up to 10 Mhz (NEW)	$ 49.95	____
____	MICRUN80 (UDI) requires V-20/30	$450.00	____
____	MICRUN80 5 MHZ Z80 Board	$595.00	____
____	MICRUN80 9 MHZ Z80 Board	$995.00	____
____	MICRUN86 (UDI)	$450.00	____

Sub Total ____

Add $5.00 shipping charge per order except for V-20/30 chips which include shipping. International orders add $10.00

Amt. Encl. ____

Payment - VISA MC MONEY ORDER C.O.D. - Add $3 dollars

Credit Card Expiration ____ / ____

Card # ____

Name ____
Address ____
City ____ State ____
Zip ____ Tel. ____ / ____

30 DAY MONEY BACK GUARANTEE ON ALL PRODUCTS. SOFTWARE IS NOT COPY PROTECTED.

Micro Interfaces Corporation
6824 N.W. 169th Street, Miami, Florida 33015
(305) 823-8088
Telex 5106004680 MICRO INTER CO
Ask About Our Intel Operating System Interfaces

OEM, VAR, Dealers, Inquiries Invited

RUN INTEL SOFTWARE ON IBM PC's

Transform an IBM PC, XT, AT, or compatible into a totally integrated Intel Micro Development System, capable of running all 8 bit ISIS-II, and 16 bit ISIS-III (UDI) software while remaining 100% MS-DOS compatible. MICRUN 80 and MICRUN 86 are Universal Development Interfaces which will allow you to increase your development capabilities with faster, more reliable and affordable PC based work stations. The MICRUN80 (UDI) utilizes the N.E.C. V-20/30 8/16 bit microprocessor to execute 8 bit Intel software on a PC, or XT, 2 to 3 times faster then an Intel MDS! For ultra fast execution speed, our Z80 co-processor version of MICRUN 80 runs in excess of 9 Mhz and is PC "AT" compatible. The MICRUN86 (UDI) is the perfect solution for running 16 bit ISIS-III software on PC's or XT's, and AT's and does not require any additional hardware. Both MICRUN80, and MICRUN86 come complete with a full featured communication program for transferring Intel software to your PC. Micro Interfaces Corporation has brought the power of Intel development to the PC, and in doing so provided you with an affordable alternative to the high cost of Intel development. To order, or obtain more information call the toll free number listed above or refer to the order coupon.

* Requires V-20 or V-30 microprocessor

RUN/CPM is a registered trademark of Micro Interfaces Corporation; Sidekick is a registered trademark of Borland International; Kaypro is a registered trademark of the Kaypro Corporation; Intel is a registered trademark of the Intel Corporation; CP/M is a registered trademark of Digital Research.

(continued from page 42)

you have to start the definition from the beginning and not from the point where the error occurred. This behavior for compilation errors is common to many Forth implementations.

It should be noted that the word that was being compiled into the dictionary at the time is in the dictionary in a smudged state. Normally this is alright because the word cannot be found and accidentally executed. If *FORGET* has been compiled, though, the partially compiled word cannot be forgotten unless *SMUDGE* is executed before any more manipulation of the dictionary occurs.

Stack errors are not announced. This is not as bad as it first sounds because both the return stack and the user stack are implemented as 128-word circular queues. Being circular, a runaway stack cannot damage anything but the stack itself. With 128 words in the stack, it's not likely that anything useful in the stack would be corrupted by a stack error. It does mean, though, that if a program pops data off an empty stack, it will continue on, using whatever it happened to find as data, without ever telling you that there was no data there in the first place.

Other error messages are not implemented. Specifically, these include:

1. Incomplete definitions—Improperly paired *DO . . . LOOP*, *IF . . . THEN*, *BEGIN . . . UNTIL*, and similar structures are not detected. Words that are compiled with incomplete structures will probably compile with no trouble, but trying to execute such a word will almost certainly crash the system.
2. Inappropriate execution—Attempts to execute words that are compile only, such as *DO . . . LOOP*, are not flagged. Trying to execute such words will crash the system.
3. Defining a word that already exists—In a normal Forth system, redefining a previously existing word causes a nonfatal warning message to be issued and compiling may continue. In this system, no message of any sort is issued and the continued compiling is allowed.

Major Deviations from the Standard

In a few cases, I deviated from the defined 79 Standard for special reasons. In the spirit of Mountain View Press' Forth implementations, one thing that I did was replace words of the form (xxx) with <xxx>. This is because the close parenthesis in (xxx) words causes trouble with Forth comments.

<NUMBER>, the word that tries to interpret input as a number, does not recognize double-precision numbers. This is a violation from the standard and was done in order to reduce the size of the system. Also, *<NUMBER>* takes an address as input and returns either a false flag (0) if the conversion fails, or it returns a true flag (hex FFFF) and an address of the result if the conversion succeeds. The Forth word *NUMBER* would be defined as the following (except for the matter of double-precision mentioned above):

```
: NUMBER <NUMBER> NOT IF ABORT"
                    NOT RECOGNIZED" THEN ;
```

The word *NOT* is defined in a nonstandard way. It returns the one's complement of the value on the stack. Normally *NOT* is a synonym for *0=*, which returns a true (hex FFFF) if a 0 is on the stack and a false (0) otherwise. This version of *NOT* will allow proper execution of standard Forth words that use *NOT* to complement the result of a logical test (. . . 9 = NOT . . .). If, however, a standard Forth word uses *NOT* when *0=* is actually meant (for readability reasons perhaps), then the program will not behave properly. This change was made because it was thought to be more useful for process-control routines to have a one's complement word directly available.

The word *BASE* is a 1-byte variable instead of the usual 2-byte form. This means that *BASE* should be accessed and manipulated by using *C@* and *C!* instead of *@* and *!*. Similarly, *>IN* is a 1-byte variable. The word *TIB* is a constant, not the usual variable, so the location of the input buffer cannot be changed.

The Future

To date (March 1986) none of the Forth-controlled floats have actually gone to sea. So far they have been used on the lab bench to evaluate the new sensors that will be on board the floats that are to go into the water this coming fall.

The lab bench floats are not in their glass tubes, so communicating with the CPU is just a matter of plugging into the I/O port (port B in the listings). With the seagoing floats, the details of the communication link are still an open issue. The link will either involve a watertight connector that goes through the glass wall or orthogonally mounted (in order to avoid interference) optical links.

A second vital use that these floats are serving is to teach the rest of the research group the Forth language. Forth has a long history in process control and data aquisition, but its use in oceanography is relatively new. Having RAFOS float CPUs with Forth ROMs readily available has proved invaluable as a teaching aid for learning Forth.

As a final footnote, I should point out that the code in Listings One and Two is the result of one person's total immersion in the implementation process. As such it could very well suffer from the lack of multiple critical perspectives on its design. I would welcome critiques of the result.

Bibliography

I have listed below a few Forth references that were my constant companions while working on this implementation. In addition, there is a short paper describing the first results from the first generation of RAFOS floats.

Forth Standards Team. *Forth-1979 Standard.* Forth Interest Group.

Haydon, G. B. *All About Forth.* MVP-Forth Series, vol. 1. (Mountain View Press Inc.).

Ritter, T. "Varieties of Threaded Code for Language Implementation." *Byte* (Sept. 1980): 206–227.

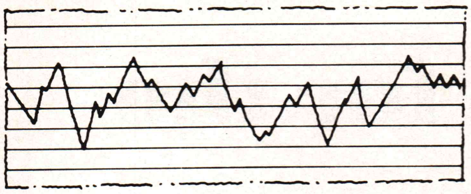
Rossby, H. T., Bower, A. S., and Shaw, P. T. "Particle Pathways in the Gulf Stream." *Bulletin of the American Meteorological Society*, vol. 66, no. 9 (Sept 1985): 1106–1110.

DDJ

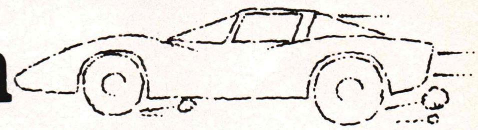
(Listings begin on page 84.)

Vote for your favorite feature/article.
Circle Reader Service **No. 7.**

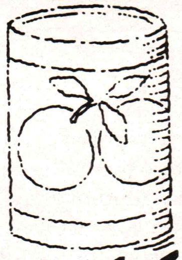
VEDIT® Plus Text Editor



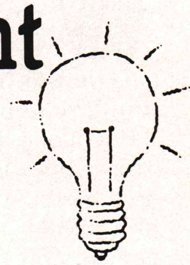
The Navy charts new concepts with it... GM engineers the future with



it...



National Can preserves facts with it... GE has bright ideas with it... Here's why you shouldn't be without it.



Every day, VEDIT PLUS helps thousands of programmers, writers and engineers get down to business.

So why do people who could have ANY text editor prefer ours? For a lot of reasons, including:

- CAPACITY—With VEDIT PLUS, file size is never a problem. And virtual disk buffering simplifies editing of even the largest files.
- FLEXIBILITY—VEDIT PLUS lets you edit up to 37 files simultaneously. So you can cut and paste. Edit programs. Edit text. Even perform numerous search/replace functions in several files without user intervention.*
- CUSTOMIZATION—With VEDIT PLUS, you can create your own on-line editing functions with keystroke macros. Develop your own on-line help screens. Determine and revise your own keyboard layout easily.
- SPEED—VEDIT PLUS not only works hard, it works fast. Faster, in fact, than any other text editor on the market.
- EXPERIENCE—Six years ago, CompuView revolutionized the concept of microcomputer text editing. And we've been improving our products and services ever since.

Special Offer: Order a VEDIT PLUS text editor for $225 and we'll include our V-PRINT™ document formatter—a $120 value—absolutely free.

Call CompuView today at 313/996-1299. You'll be in good company.

CompuView®

CompuView® Products Inc., 1955 Pauline Boulevard—Suite 300, Ann Arbor, Michigan 48103, TELEX 701821

Available for PC DOS, MS-DOS, CP/M, CP/M-86.

*Free sort, compare, print and main menu macros included; optional 8080-8086 translator or mailmerge, $50 each.

Circle no. 122 on reader service card.

Forth Windows for the IBM PC

by Craig A. Lindley

The demo program shows how to integrate the window package with an application.

Windows provide a method of presenting information to computer users in an easy-to-use, natural manner. The window environment can be thought of as an emulation of a desk. The analogy is that you are working on something that resides on the top of a pile of paper. If you are interrupted, new work is placed on that pile, covering up the work you were doing. When you finish with this new work, you move it off your desk and continue where you left off on the work you were doing before. This interruption and restoration of the working environment is the concept on which the window software metaphor is based.

This article illustrates the use of windows in the Forth environment. The window package I present here uses the F83 dialect of Forth developed by Laxen and Perry. The program is based upon and inspired by the article entitled "A Simple Window Package," by Edward Mitchell, which appeared in the January 1984 issue of *DDJ*. In addition to the primary topic of windows, I'll also discuss some other subjects, including MS-DOS memory management, manipulation of the IBM PC hardware, interfacing Forth to the PC's BIOS and BDOS, and Forth 8088 assembly language.

People with computers other than an IBM PC (or a true compatible) and people using other dialects of Forth may need to make some changes to the program in order to get it to work on their systems. The concepts presented in this Forth implementation of windows, however, can be applied to an implementation in any other computer language, including C or Pascal.

The program presented in Listing One, page 96, is basically a tool awaiting your use; it is meant to be integrated into an application program. The number of windows you can have on your display screen at one time is a function only of the amount of memory you have in your computer—you are not limited by the available memory in the 64K segment in which Forth is running. The demonstration program provided in the listing shows how to integrate an application (the demo itself) with the window routines.

Memory Management

It's important to understand how MS-DOS performs memory management if you are to understand how the window package works and why it isn't shackled by the 64K segment constraint imposed on the Forth system. Memory management is not necessary just because I'm using Forth for this program; it's necessary for any program, in any language, that uses more than 64K in its operation.

Upon receiving control from MS-DOS, an executable program is given all the memory available in your computer for its use, whether or not your program requires this much memory. Under these conditions MS-DOS cannot manage memory because there is none left to manage—it all belongs to your executable program. In order for MS-DOS to manage the memory, your program must give back to the operating system the memory it doesn't need. This is done by using the *setblock* function of MS-DOS. By informing MS-DOS of the total memory requirement of your program, your allocated memory block will shrink and the MS-DOS pool of free memory will be given the remainder of the available memory in your computer. This free pool of memory can then be managed for your uses by MS-DOS.

Three special words in the Forth window package deal with MS-DOS memory management. They are *setblock*, *calloc*, and *free*. *Setblock* shrinks the memory block allocated to the Forth program of which it is a part. The *setblock* word defined on screen 7 accepts as a parameter the number of bytes required by the application (the Forth system). It returns a true flag if the memory block size adjustment was successful, or it returns a false flag, error code, and the maximum number of 8088 paragraphs available if the adjustment was unsuccessful. The error codes correspond to those listed in the DOS manual.

In my window application, the word *initialize* calls *setblock* and passes it a −1 (FFFF hex) as the number of bytes to be set aside for Forth's

Craig A. Lindley, 6 Sutherland Pl., Manitou Springs, CO, 80829

use. In other words, Forth is given a full 64K segment in which to run, thereby excluding MS-DOS memory management from that area. Forth owns all of this 64K block.

After *setblock* gives MS-DOS some memory to manage, calls to the Forth word *calloc* cause MS-DOS to provide a memory area for your use. This memory area comes directly out of the free pool and belongs to the application until it is given back. The word *calloc* accepts as a parameter the number of bytes required for a new memory block. It returns a true flag and the 8088 segment address if successful, or it returns a false flag, error code, and maximum number of paragraphs available from the free pool, if the allocation fails.

Just as memory blocks are given out by MS-DOS, they can also be given back when an application has finished with them. The Forth word *free* performs that function in the window package. Any memory block that has been allocated previously can be released. The word *free* accepts the segment address of the block to be returned and returns either a true flag if successful or a false flag and error code if not.

You should keep two things in mind when dealing with MS-DOS memory management functions. First, a program should never write into any memory it does not explicitly own—for example, programs must be sure the stack is in an area of memory owned by the application. Second, never try to release memory that was not allocated by MS-DOS originally. Violating either of these rules can result in unexplained behavior on the part of your computer.

The memory management words used in this window program all make calls to MS-DOS *int 21h* using the appropriate function codes. MS-DOS performs the requests, if possible, and the Forth words pass back flags on the stack indicating the status of the requested operation.

Low-Level Assembly-Language Words

Approximately one third of all the Forth words in this window package are written in 8088 assembly language (see Tables 1 and 2, pages 47 and 48). There are two reasons for this. First, the F83 package used to develop this

chra   char/attrib count --
Writes (count) characters with attributes to the screen starting at the current cursor position. The cursor position is left unchanged. This word calls BIOS *int 10h* using function code *9* to perform its operation.

chra+   char/attrib --
Similiar to *chra* except only a single character with attribute is written and the cursor position is advanced automatically. BIOS *int 10h* is again used to first write the character and attribute and then to read and advance the cursor position.

rdchra   -- char/attrib
Calls BIOS *int 10h* with function code *8* to return the character and attribute of the character currently under the cursor.

scrlup   xul yul xlr ylr count attrib --
Scrolls up the area of the screen bounded on the upper left by *xul* and *yul* and on the lower right by *xlr* and *ylr.* The window is scrolled up (count) lines, and the blank lines scrolled in from the bottom are given attribute (attrib). If count is specified as *0*, the whole window is cleared.

calloc   # of bytes -- seg T -- max paragraphs error code F
Initiates a memory allocation request to MS-DOS. If the memory requested is available, the segment address and a true flag will be returned. If enough memory is not available, then an error code and a false flag are returned along with the maximum number of paragraphs available.

free   seg -- T -- error code F
Attempts to release to MS-DOS a block of memory previously allocated via *calloc* If successful, a true flag is returned. If unsuccessful, an error code and a false flag are returned.

setblock   # of bytes -- T -- max paragraphs error code F
Asks MS-DOS to shrink or expand the unassigned memory until the application program has the number of bytes requested for its use. If the operation is successful, a true flag is returned. If not, an error code and a false flag are returned, along with the maximum number of paragraphs available.

e@   seg addr -- n
Returns to the top of the parameter stack the data (*n*) at address (*addr*) in memory segment (*seg*). I call this word extended fetch because it has access to the complete memory space and not just the 64K segment in which Forth runs.

e!   n seg addr --
Stores the data (*n*) at address (*addr*) in memory segment (*seg*). I call this word extended store because it has access to the complete memory space and not just the 64K segment in which Forth runs.

rdcur   -- x y
Uses BIOS *int 10h* function code *3* to return the x,y location of the cursor.

save_h, save_w, save_ptr, save_si, save_ds   -- addr
These Forth words are used as temporary storage locations during the *scr->buf* and *buf->scn* routines. When executed, they return the address of a 2-byte storage area. The storage area *save_h* is used to save the height parameter, *save_w* the width parameter, *save_ptr* the address in screen memory to which data is to be saved or restored, *save_si* the 8088 *si* register that is used by Forth as the instruction pointer, and *save_ds* the data segment in which Forth is running.

scn->buf   x y width height seg --
Moves memory a word at a time from the appropriate position in the screen memory to a buffer defined by the seg parameter. Both the character and attribute residing on the screen at a given location are moved. X and y mark the upper-left corner of the rectangle to be moved.

buf->scn   seg x y width height --
Moves memory a word at a time from a buffer in memory defined by (*seg*) to the appropriate position in the screen memory. Both the character and attribute stored in this memory buffer are moved. X and y mark the upper-left corner of the rectangle to be restored.

Table 1: *Forth assembly-language word definitions*

FORTH WINDOWS
(continued from page 47)

software has limited BIOS/BDOS support for the special functions required. Second, assembly-language code executes faster than any higher-level language, including Forth.

Three different levels of software interface are used in this window package. The lowest level, which involves direct manipulation of the PC hardware, includes the words *buf->scn* and *scn->buf*. Both of these Forth words read and/or write to video RAM directly while saving and restoring of the display is taking place. Because of their direct manipulation of the video display, these words are also the least transportable. They must know whether they are running in a PC that has only a monochrome monitor or one with a graphics adapter. This is done by changing the value of the constant *v_seg* in the window program and recompiling. The correct values of *v_seg* are as follows:

color graphic adapter *v_seg* = B800h
monochrome monitor *v_seg* = B000h

The constant *v_seg* informs both *scn->buf* and *buf->scn* of where the video memory begins, and they do the rest. See the section on the save and restore algorithm for specifics on how these words work.

The next higher level of software interface to the PC makes use of the BIOS routines. In this program, extensive use is made of the functions provided by the video BIOS interrupt, *int 10h*. The video functions are accessed by placing parameters in the various 8088 registers, placing a function code in the *ah* register, and issuing the *int 10h* request. Table 3, page 50, gives a summary of which *int 10h* functions are used and where.

When using the BIOS functions, it's important to save any registers of special significance as many of the BIOS routines alter registers during the course of their operation. This version of Forth, for example, uses the 8088 *si* register as the instruction pointer and the *bp* register as the return stack pointer. These registers, therefore, must be saved and restored after any BIOS routines that modify them are used. In all the low-level word definitions that access the BIOS routines, you'll see *si push* and *si pop* instructions surrounding the BIOS interrupt call. You'll also find a *bp push* and *bp pop* in the *scrlup* word definition because the *bp* register is modified there.

The highest level of software interface is on the BDOS level. The memory management words *calloc, free,* and *setblock* are examples of this

case, of, endof, and *endcase*
Dr. Charles Eaker's *case* statement. See *Forth Dimensions*, vol. II, no. 3, p. 37 for details on how these words work.

putch x y char/attrib --
Writes the character and attribute onto the video display screen at location x,y. The cursor position is set at the next character.

getch x y -- char/attrib
Returns the character and attribute at location x,y on the video display screen. The cursor is moved automatically to position x,y.

draw_row x y char/attrib count --
Displays (count) identical characters starting at the position on the video display screen defined by x,y.

ulx, uly, width, height, curx, cury,
oldx, oldy, bufseg, oldwcbseg, attrib -- n
These words are constants that define the positions of storage locations within the current window control block (wcb). When executed, they return offsets relative to the start of the wcb storage area. (See Table 7.)

wcbseg! n addr --
Stores information into the active wcb. The active wcb is the one whose segment address is in the variable *wcbseg*. For example, *7 attrib wcbseg!* would store the display attribute 7 into the *attrib* slot in the active window control block. The *attrib* word supplies the address in which to store the display attribute.

wcbseg@ addr -- n
Fetches information from the active wcb. For example, *attrib wcbseg@* would fetch the display attribute from the active window control block and put it on the parameter stack.

top, sides, bottom --
These words draw the actual window frame on the display screen. When executed, they draw the top, sides, and bottom, respectively, of the window specified in the active wcb. The program constant *border* shown in the listing is used to determine the attribute with which to draw the window frame. The current version sets it to high-intensity normal video.

((window)) --
This is the lowest-level window routine. It automatically fetches from the current wcb the position and size of the window to be drawn on the display, copies to the appropriate window buffer the portion of the display screen that will be overwritten by this new window, and then draws the window frame by invoking *top, sides,* and *bottom.*

clr_window --
Clears the current window by fetching all the appropriate parameters from the wcb and invoking *scrlup* to clear the entire window. It then sets the window cursor position *curx, cury* to *0* to home the cursor in the window.

(window) x y width height attrib -- f
Builds the actual window. It tries to allocate enough memory to hold a new wcb (22 bytes). If successful, it links this new wcb into the wcb linked list and then tries to allocate enough memory to contain the screen information that the new window will overwrite. If this, too, is successful, all the parameters passed to this routine are stored in the new wcb, *((window))* is called to draw the actual window, and a true flag is returned to the calling program indicating that the creation of the window was successful. If either allocation attempt fails, the memory previously allocated is freed and a false flag is returned indicating win-

Table 2: *High-level Forth words*

BDOS interface. These words work by loading parameters into the 8088 registers, loading a function code into the *ah* register, and executing *int 21h*. All functions provided by *int 21h* save and restore all registers (except those used to pass back parameters), so the precautions used for the BIOS routine interface aren't required. Table 4, page 50, summarizes memory management functions performed via the BDOS interrupt.

dow creation failure. Under these conditions, an error message will be displayed to help the programmer find the source of the problem. In most cases a failure indicates lack of available memory.

open_window x y width height attrib -- f
This is the highest-level window word. Its function is to perform checking on the specified window parameters to verify validity. If the specified window wouldn't fit on the display screen, an appropriate error message will be displayed. Another error message will be displayed if the proper parameters are not present on the parameter stack. This word will not allow the programmer to create a window that cannot be displayed on the screen correctly.

close_window --
Closes the current window. A window is closed by moving the screen data stored in the memory buffer back onto the display screen, then freeing the memory allocated to both the wcb and the memory buffer. Next, the cursor is returned to where it was before this window was opened, and then the wcb is removed from the wcb linked list. If no windows are currently open, execution of this routine will result in an error message.

wat x y --
This routine (pronounced "window at") places the cursor in the window at the location specified by the x,y coordinates. These coordinates are relative to the current window, not the whole display screen. If either coordinate exceeds the size of the current window, the cursor will be placed as close as possible to the position specified without leaving the window.

rdwcur -- x y
Returns the cursor position relative to the current window.

rdwcha x y -- char/attrib
Returns the character and attribute found under the current window's cursor.

scroll_window --
Scrolls the current window up by one line to allow new information to be displayed. This word gets all the parameters it needs from the wcb. The attributes used for the blank line scrolled onto the display are the same as those specified when the window was created.

crout, lfout, bsout, bell --
These words perform, in the current window, the same function as they would perform if issued to the normal screen. Namely, they return the cursor to the first character position of the current line, move the cursor down one line, back the cursor up by one character position, and cause the computer to ring its chimes, respectively. Special versions of these functions are required to keep the cursor within the window.

wemit char --
This word (pronounced "window emit") is the equivalent of the Forth word *emit*. It should be used only when writing information to windows. In addition to sending normal characters to the display window, it performs the *cr, lf, bs,* and *bell* functions as described above. If a line feed (*lf*) character code is issued while the window cursor is on the last line of the window, *wemit* will scroll up all the information in the window accordingly.

initialize --
Sets up the MS-DOS memory management function. It requests a full 64K segment for the Forth system currently running. If this initialization is successful, an appropriate message is displayed and the *wcbseg* variable is set to *0*, indicating that no windows are currently active. If there wasn't enough memory, an error message is displayed and the window program is completely aborted.

Table 2 (continued): *High-level Forth words*

Forth 8088 Assembly Language

Assembly language in Forth is a bit unusual. The first conceptual hurdle that needs to be overcome involves the reverse Polish or postfix notion used throughout the assembly-language definitions. Assembly language written in this manner has the operand(s) preceding the mnemonics. For example, the standard 8088 assembly-language instruction *mov dh,dl*, where *dh* is the destination and *dl* is the source, is coded in Forth as *dh dl mov*.

Some new symbols are also necessary to allow the Forth assembler to create the correct instruction sequences. Special symbols used in this program are *#*, which indicates an immediate operand, and *#)*, which indicates an indirect operand. These symbols are sprinkled liberally throughout the low-level Forth code words.

A few other special-purpose Forth words are used in assembly-language definitions. The word *code* takes the name that follows it and creates a new dictionary entry for the code definition to follow and sets up certain pointers for Forth so that this code definition can be executed in a manner similiar to that of all Forth definitions. *Code* also tells the compiler to reference the assembler vocabulary so that all 8088 mnemonics that make up the definitions can be found in the dictionary searches and assembled into the code.

The word *next* (and its derivative *1push*) causes a direct jump back to the Forth inner interpreter. This restores control to the interpreter, which then passes control to the next Forth word in the program. Writing a code definition and forgetting to have *next* or *1push* as the final statement (before *end-code*) will cause your computer to crash right after the word is executed.

The word *1push* performs the same function as *next* does, except it pushes the contents of the 8088 *ax* register onto the parameter stack before returning to the inner interpreter. In this way, parameters can be passed back via the stack to subsequent high-level Forth definitions.

The final word *end-code* changes the vocabulary back to Forth (from assembler), performs a limited

(continued from page 49)

amount of error checking on the code definition, makes this new word visible in the dictionary, and then terminates the assembly-language word definition.

From the listing you can see how easily assembly-language word definitions can be integrated into a high-level Forth program. From my experience with in-line code written in any computer language, the following rules apply:

- Use in-line code only when speed of execution is required.
- Keep the definitions as small as is practical. This will aid in debugging and maintenance.
- Code carefully; errors in in-line code can send your computer on a journey from which it will never return except via reboot.

Screen Save and Restore Algorithm

The screen save and restore algorithm was developed to keep to a minimum the number of calculations necessary to locate the correct screen data and move it to/from buffers allocated for its storage. Minimal calculations result in the fastest possible execution of the windows.

All moves of data to or from the screen and buffers are word moves because the character and attribute for a screen character are stored in sequential bytes of screen memory. Character data is stored at even addresses, and attribute data is stored at odd addresses. When a word is fetched from an even addess in screen memory, the high byte contains the attribute and the low byte contains the actual character. For the same reason, sequential lines on the video display are offset not by 80 bytes but rather by 160 bytes. This fact is important whenever data is moved to or from the PC screen and is taken into consideration by the *scn->buf* and *buf->scn* word definitions.

A program design language (PDL) representation of the data movement algorithm is shown in Table 5, right. The start address of the video data needs to be calculated only once before the nested loops actually perform the work of moving the data. The 8088 block-move word instructions make the data movement from screen memory to buffer and from buffer back to screen memory a trivial task. The incrementing of the pointers shown in the PDL is handled automatically by specifying block-move word instructions. For both directions of data movement, the *ds:si* register pair points at the source of the data to move, and the *es:di* register pair points at the destination. Both *si* and *di* increment automatically by 2 for movement of word-size data.

Windows in Your Programs

Interfacing windows with your application program is relatively straightforward once you understand how the window package works. Once you have compiled the package, you can create windows interactively to see how they work before trying to use them in your program. For example, typing the commands

```
initialize 0 0 20 10 7 open_window
```

at the keyboard will immediately create a window with its upper-left corner at 0,0; a width of 20 characters; and a height of 10 vertical lines. (Width and height are inside dimensions of the windowed area; the bor-

Function	Function Code	Used in Forth Word(s)
write char with attribute	9	*chra, chra+*
get cursor position	3	*chra+, rdcur*
set cursor position	2	*chra+*
read char and attribute	8	*rdchra*
scroll up video window	6	*scrlup*

Table 3: *Summary of* int 10h *functions*

Function	Function Code	Used in Forth Word
allocate memory block	48h	*calloc*
release a memory block	49h	*free*
resize an allocated block	4ah	*setblock*

Table 4: *Memory management functions performed via the BDOS interrupt*

```
procedure scn->buf parameters are x, y, width, height and buf seg
begin
    set data movement direction to forward
    get buffer segment address into the extra segment reg.
    initialize destination ptr DI to 0 which points at the first byte of buffer storage
        in the buffer segment.
    save the height parameter in temporary storage
    save the width parameter in temporary storage
    get y coordinate of first location to store
    multiply by 160 to find line start address
    get x coordinate of first location to store
    multiply by 2 to find character address offset
    add new x and new y to get start address of data move
    save result in a temporary location called save_ptr
    save the current value of the SI register in a safe place
    save the current value of the DS register in a safe place
    move video segment address v_seg into the DS register
    do height times
        do width times
            move DS:SI to ES:DI  (move actual data)
            increment SI and DI both by 2 for word moves
        enddo
        save_ptr = save_ptr + 160  (move down 1 vertical line)
    enddo
    restore previous DS register value
    restore previous SI register value
    jump to next   (back to inner interpreter)
end
```

Table 5: *PDL representation for the data movement algorithm*

der of the window will take up two more characters horizontally and two lines vertically.) The attribute code of 7 will create the window using normal, low-intensity video. Remember, the word *initialize* is needed to set up the MS-DOS memory manager. If *initialize* is not executed before the *open_window* request, the request will fail.

Once a window is opened, you can move the window's cursor to any position by using the word *wat* (the window counterpart of the Forth word *at*). You can write text into the window using *wemit* and *wtype* (the window counterparts of *emit* and *type*). The special video control codes shown in Table 6, below, are also supported via *wemit*. Note that by redefining the F83 deferred word *emit* to *wemit*, you can get Forth to run in one of its own windows.

All words that write text into a window (such as *wemit*) always work on the currently selected window only [the window at the end of the window control block (wcb) linked list whose wcb address is contained in the Forth variable *wcbseg*]. Thus, continuing with the example, if you were to open a second window, *wemit* would then automatically write to this new window. The previous window could not be written to again until the new window was closed. The Forth word *close_window* closes the current window and reopens the previous window if one exists. If *close_window* is executed when no windows are open, an error message is displayed. A window that is closed erases itself from the screen (by restoring the screen data that it covered up), frees the memory it had allocated for the window control block and the screen buffer, and finally unlinks itself from the wcb list. Table 7, below, shows the structure of an entry in the wcb list.

The demo program in the listing is an example of how an application program can be integrated with the basic window package. It demonstrates opening windows using various attributes, writing text to the windows, using the special video control codes, listing Forth screens in a window, linking overlapping windows, clearing windows, and closing them. It illustrates how easy to use and how fast the windows can be.

DDJ

(Listing begins on page 96.)

Vote for your favorite feature/article.
Circle Reader Service **No. 8.**

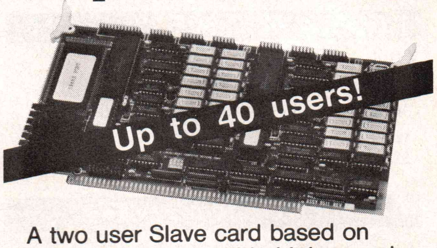
Video Code	Function
7	Ring the PC's bell
8	Backspace the window cursor
10	Linefeed (scroll upwards if necessary)
13	Carriage return

Table 6: *Video control codes*

Displacement	Name	Description
+ 0	ulx	X coordinate of upper left corner
+ 2	uly	Y coordinate of upper left corner
+ 4	width	Width of the window
+ 6	height	Height of the window
+ 8	curx	Window—relative cursor position
+10	cury	Window—relative cursor position
+12	oldx	Cursor position in previous window
+14	oldy	Cursor position in previous window
+16	bufseg	Points to window's text buffer
+18	oldwcb	Points to previous wcb record (NIL for last record)
+20	wattrib	Window's text attribute byte

Table 7: *Structure of an entry in the window control block list*

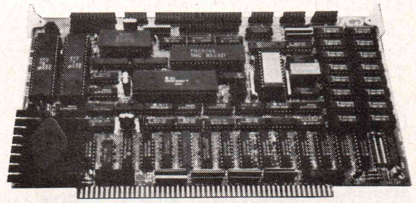
Products With Expandability



A two user Slave card based on Hitachi's Z80 compatible high speed, 10MHz super microprocessor.

Priced at **$495.00***

Features Include...
- 4-10 MHz Z80 Compatible HD64180
- 1/2 Megabyte Nonbanked Memory
- 2 Asynchronous Serial Ports To 38.4
- 1 High Speed Synchronous Port
- All Transfers Via 1.6 MHz DMA!!!
- Unique Expansion Port Offers;
  2 Additional Serial Ports or . . .
  2 Parallel Ports or . . .
  Real Time Clock With Battery Backup



The industry's fastest 8-bit Master CPU card with features superior to most 16-bit cards.

Priced at **$495.00***

Each Master Features . . .
- 4-10 MHz Z80 Compatible HD64180
- 1/2 Megabyte Nonbanked Memory
- 2 Asynchronous Serial Ports To 38.4
- 1 High Speed Synchronous Serial Port
- 4 Bi-directional Parallel Ports
- TurboDOS**, ZSYSTEMS**, CP/M**, & OASIS** Operating Systems
- FDC Simultaneously Controls 8", 5¼", & 3½" Drives
- SASI/SCSI Interface
- Optional High Speed Hard Disk/File Access Tape Backup and True ETHERNET Controller

**Prices apply to 6 MHz, 64KB versions and are good for a limited time only on purchases of ten or more. For less than ten, please call.*

***Trademarks: TurboDOS - Software 2000; ZSYSTEMS - Echelon; CP/M - Digital Research; OASIS - THEOS Software*

ICD INTELLIGENT COMPUTER DESIGNS CORP.
23151 Verdugo Drive, Suite 113
Laguna Hills, CA 92653
(714) 581-7500

Circle **no. 278** on reader service card.

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

Artek ADA Compiler - DoD Standard minus multitasking. Includes trace, breakpoints, code disassembly, more. Produces virtual stack file or native code. PC $ 895

AI-Expert System Dev't

Arity System - incorporate with C programs, rule & inheritance PC $ 295
Experteach - Powerful, no limit on memory size. Samples PC $ 399
EXSYS - Improved. Debug, file & external program access. MS $ 339
Insight 2 - dB2, language. MS $ 399
Others: APES ($359), Advisor ($949), ES Construction ($100), ESP ($845), Expert Choice ($449)

AI-Lisp

BYSO - Common, MacLISP compatible. 250+ functions, fast PC $ 150
GC LISP Interpreter - "Common", rich. Interactive tutorial Call
Microsoft MuLisp 85 $ 199
TLC LISP - "LISP-Machine" - like, all RAM, classes, **compiler.** MS $ 225
TransLISP - learn fast MS $ 75
WALTZ LISP - "FRANZ LISP" - like, big nums, debug, CPM-80 MS $ 149
Others: IQ LISP ($155), UNX LISP ($59), IQC LISP ($269)

AI-Prolog

ARITY Standard - full, 4 Meg
Interpreter - debug, C, ASM PC $ 350
COMPILER/Interpreter-EXE PC $ 795
With Exp Sys, Screen - KIT PC $1250
MicroProlog - enhanced MS $ 229
Professional MicroProlog MS $ 359
Prolog-86 - Learn Fast, Standard, tutorials, samples MS $ 95
Prolog-86 Plus - Develop MS $ 250
TURBO PROLOG by Borland PC $ 85
Others: Prolog-I ($365), Prolog-2 ($1795)

AI-Other

METHODS - SMALLTALK has objects, windows, more PC $ 215
QNIAL - Combines APL with LISP. Source or binary. PC $ 359

FEATURES

Synergy - Create user interfaces. TopView - compatible multitasking operating environment offers windows, icons, pull-down menus and fonts in 12K RAM. MS $ 375
Microsoft Cobol Tools - symbolic, windowing debugger w/source support. Plus cross reference, menu handler, mouse support. Xenix. $359 PC $259

Compare Products Call Product Specialists

The Programmer's Shop is much more than the supplier of the largest selection of programmers' software. Trained programming consultants will answer your questions. Ask "product specialists" about Cross Assemblers, Translators, Debuggers, or C compilers.

Our Services:

- Programmer's Referral List
- *Compare Products*
- Help find a Publisher
- Evaluation Literature FREE
- BBS - 7 PM to 7 AM 617-826-4086
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

Basic

ACTIVE TRACE Debugger - BASICA, MBASIC, well liked MS $ 79
APC MegaBASIC - powerful PC $ 339
Basic Development System - for BASICA; Adds Renum, more. PC $ 105
Basic Windows by Syscom PC $ 95
BetterBASIC - all RAM, modules, structure. **Full BASICA** PC $ 169
8087 Math Support PC $ 89
Run-time module PC $ 235
Better Tools - for Better Basic PC $ 95
CADSAM FILE SYSTEM - full ISAM in MBASIC source. MS $ 75
Prof. Basic - Interactive, debug PC $ 79
8087 Math Support PC $ 47
QuickBASIC by Microsoft - Compiles full IBM BASICA, 640K PC $ 79
TRUE Basic - ANSI PC $ 119
Run-time Module PC $ 459

Cobol

Macintosh COBOL - full MAC $ 459
MBP - Lev. II, native MS $ 885
MicroFocus Professional - full PC Call
Microsoft Version II - upgraded. Full Lev. II, native, screens. MS $ 495
Realia - very fast MS $ 889
Ryan McFarland COBOL MS $ 699
COBOL-8X MS $1100

Editors for Programming

BRIEF Programmer's Editor - undo, windows, reconfigure PC Call
C Screen with source 80/86 $ 75
EMACS by UniPress - powerful, multifile, windows Source:$949 $299
Epsilon - like EMACS, full C-like language for macros. PC $169
Kedit - like XEDIT PC $109
Lattice Screen Editor - multiwindow, multitasking Amiga $100 MS $109
PMATE - power, multitask 80/86 $159
VEDIT - well liked, macros, buffers CPM-80-86 MS $119
XTC - multitasking PC $ 85

Atari ST & Amiga

We carry full lines of Manx, Lattice, & Metacomco.
Cambridge LISP Amiga $ 200
Lattice C ST Amiga $ 139
Lattice Text Utilities Amiga $ 75
Megamax - tight, full ST $ 179

RECENT DISCOVERY

Microsoft Windows Software Development Kit - Run graphics programs on all machines that support Microsoft Windows. Eliminates need for hardware - specific support. Includes MS Windows user version PC $399

C Language-Compilers

AZTEC C86 - Commercial PC $499
AZTEC C65 - Personal Apple II $199
C86 by CI - 8087, reliable MS $299
Lattice C - from Lifeboat MS $289
Lattice C - from Lattice MS $339
Mark Williams - w/debugger MS $399
Microsoft C 3.0 MS $259
Q/C 88 by Code Works - Compiler source, decent code, native MS $125
Wizard C - Lattice C compatible, full sys. III, lint, fast. MS $389

C Language-Interpreters

C-terp by Gimpel - full K & R MS $249
INSTANT C - Source debug, Edit to Run-3 seconds MS $389
Interactive C by IMPACC Assoc. Interpreter, editor, source, debug. PC $225
Introducing C - self paced tutorial PC $109
Run/C Professional - Run/C plus create add-in libraries, more MS $189
Run/C Lite - improved MS $109

C Libraries-General

Application Programmer's Toolkit MS $375
Blaise C Tools 1 ($109), C Tools 2 $ 89
C Essentials - 200 functions PC $ 85
C Food by Lattice-ask for source MS $109
C Utilities by Essential - Comprehensive screen graphics, strings, source. PC $139
C Worthy Library - Complete, machine independent, source MS $295
Entelekon C Function Library PC $119
Entelekon Superfonts for C PC $ 45
Greenleaf Functions - portable, ASM $139
PforCe by Phoenix - objects PC $299

C Libraries-Communications

Asynch by Blaise PC $149
Greenleaf - full, fast PC $139
Software Horizons - pack 3 PC $119

C Libraries-Files

FILES: C Index by Trio - full B + Tree, vary length field, multi compiler
/File is object only MS $ 89
/Plus is full source MS $349
C to dBase - with source MS $139
CBTREE - sequential, source, no royalties MS $ 99
dbVISTA - full indexing, plus optional record types, pointers, Network.
Object only - MS C, LAT, C86 $159
Source - Single user MS $429
Source - **Multiuser** MS $849
dBASE Tools for C PC $ 79

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

Circle **no. 141** on reader service card.

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

LATTICE BARGAINS ORDER TODAY

Curses - Full screen output to run on PC, UNIX, and XENIX without source changes. Compatible with UNIX Curses. PCDOS.
dbc II or III - Isam file management. Create, access, and update files compatible with dBASE II or III. PCDOS.
Text Management Utilities - 70 functions for screen, windows, pointers, and task control. PCDOS.

Order before 7/31/86 and mention this ad for the special prices below:

	LIST	Before 7/31	After 7/31
Curses	125	89	109
Curses (w/source)	250	169	219
dbc II or III	250	169	219
dbc II or III (w/source)	500	339	439
Text Utilities	120	85	105

C Support-Systems

Basic-C Library by C Source PC $139
CPRINT - by ENSCO MS $ 45
C Sharp - well supported. Source, **realtime**, tasks, state system MS $600
C ToolSet - DIFF, xref, source MS $ 95
The HAMMER by OES Systems PC $179
PC LINT - Checker. Amiga $89 MS $119
SECURITY LIB - add encrypt to MS C, C86 programs. Source $250 PC $125

C-Screens, Windows, Graphics

C Power Windows by Entelekon PC $119
ESSENTIAL GRAPHICS - fast, fonts, no royalties PC $219
GraphiC - new color version PC $319
Topview Toolbasket by Lattice PC $209
View Manager for C by Blaise PC $219
Vitamin C - screen I/O PC $139
Windows for C - fast PC $159
Windows for Data - validation PC $239

Debuggers

Advanced Trace-86 by Morgan Modify ASM code on fly. PC $149
CODESMITH - visual, modify and rewrite Assembler PC $109
C SPRITE - data structures PC $139
Periscope I - own 16K PC $269
Periscope II - symbolic, "Reset Box," 2 Screen PC $129
Pfix-86 Plus Symbolic Debugger by Phoenix - windows PC $289
Software Source by Atron - Lattice, MS C, Pascal, Windows single step, 2 screen, log file. MS $115
w/Breakswitch $199

FEATURES

dBASE Tools for C - incorporates C functions to extend III+ PC $ 79
dBASE Graphics for C PC $ 79
ZView - Screen generator gives range checking, security level settings, help tied to field or screens, dynamics. Auto convert to/from ASCII screen. MS $245

Fortran & Supporting

ACS Time Series MS $449
Forlib + by Alpha - graphics and file routines, comm. MS $ 59
MACFortran by Microsoft MAC $229
MS Fortran link to C MS **$219**
No Limit - Fortran Scientific PC $119
PolyFortran - xref, pp, screen MS $149
Prospero - '66, reentrant MS $349
RM/Fortran - enhanced "IBM Professional Fortran" MS $395
Scientific Subroutines - Matrix MS $149
Statistician by Alpha MS $269
Strings and Things - register, shell PC $ 59

Multilanguage Support

BTRIEVE ISAM MS $199
BTRIEVE/N-multiuser MS $469
CODESIFTER - Profiler. MS $109
HALO Graphics - 115 + devices. Animation, engineering, business. Any MS language, Lattice, C86 PC $239
PANEL - Create screen with editor, generates code. Full data validation, no royalties. Xenix $539, MS $239
PLINK 86-program-independent MS $279
PLINK-86 PLUS - incremental MS $369
Pfinish Performance Analyzer PC $279
PolyLibrarian by Polytron MS $ 85
PVCS Version Control MS $359
Rtrieve - Btrieve front end MS $ 79
Screen Sculptor - slick, thorough, fast, BASIC, PASCAL. PC $ 99
Xtrieve - organize database MS $169
ZAP Communications - VT 100, TEK 4010 emulation, full xfer. PC $ 85

Turbo PASCAL & Supporting

BORLAND: Turbo 3.0 $ 49
3.0 with 8087 and BCD $ 85
Turbo Graphix - graphs, windows $ 39
Turbo Toolbox or Editor $ 55
TURBO . . . Asynch by Blaise, full $ 85
Extender - "big model" $ 69
Power Tools by Blaise - library $ 85
Power Utilities - profiler, pp $ 85
OTHERS: Screen Sculptor ($99), Pascal Pac ($100), Tidy ($45).

RECENT DISCOVERY

PS Scheme LISP - by TI. Scheme has special, simple, "orthoganal" syntax. Lexical scoping, block structure, call by value, and tail-recursive semantics. Compiler, EMACS-like editor, DOS. Can support debugging graphics, windowing. PC $ 95

Other Languages

APL*PLUS/PC PC $ 469
CLIPPER-dBASE Compiler MS $ 449
HS/FORTH - '79 & '83 Standards, full RAM, ASM, BIOS, interrupts. Graph, multi-task, optimizer MS $ 250
MacASM - fast MAC $ 99
MasterForth MAC or PC $ 125
Microsoft MASM - faster MS $ 109
Microsoft PASCAL - faster MS $ 199
MICROTEC PASCAL - 5 memory models, "Iterators". 65 bit 8087 strings. MS $ 665
Modula-2/86 Compiler by Logitech w/ 8087 ($105), 512K ($149). PC $ 65
Pasm - by Phoenix MS $ 219
Prospero Pascal - full ISO + MS $ 349
RPG II by Lattice PC $ 639
SNOBOL4 + - great for strings MS $ 85

Xenix-86 & Supporting

Basic - by Microsoft $ 279
Cobol - by Microsoft $ 795
Fortran - by Microsoft $ 399
PANEL Screen LIB - multi-language $ 539
Xenix Complete Development System $1195

Other Products

Dan Bricklin's Demo Program PC $ 75
dBrief - Customize BRIEF for dBASE development. with BRIEF $275. PC $ 95
HTest/H Format - XT Fix PC $ 89
Interactive Easyflow-HavenTree PC $ 139
LMK - like UNIX make PC $ 149
Microsoft Windows PC $ 75
Opt Tech Sort - sort, merge MS $ 119
PMaker - by Phoenix PC $ 139
Polymake by Polytron MS $ 79
PolyWindows Dev. Kit PC $ 149
PS MAKE MS $ 129
Qwik Net - critical path, 125 tasks/ resources, thorough; usable PC $ 316
SECRET DISK by Lattice PC $ 49
SoftEst - Manage projects. MS $ 350
Texsys - control source MS $ 89
Visible Computer: 8088 - Simulates demos or any .exe. com, Debugger. 350 pg. tutorial, unprotected PC $ 75

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. All formats available. UPS surface shipping add $3/item.

Call for a catalog, literature, advice and service you can trust

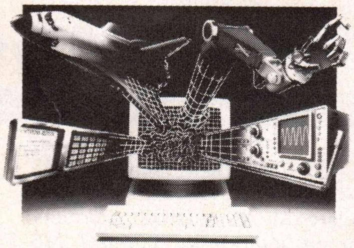
NEW HOURS
8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™
128-D Rockland Street, Hanover, MA 02339
Mass: 800-442-8070 or 617-826-7531 5/86

"I like your straightforward, open evaluations, comments and selection."
Chris Chapman
Practicle Solutions Software

polyFORTH GETS YOUR PROGRAM FROM CONCEPT TO REALITY 4 TO 10 TIMES FASTER



THE ONLY INTEGRATED SOFTWARE DEVELOPMENT PACKAGE DESIGNED FOR REAL-TIME APPLICATIONS

If you're a real-time software developer, polyFORTH can be your best ally in getting your program up and running on time. In fact, on the average, you will develop a program 4 to 10 times faster than with traditional programming languages.

polyFORTH shortens development time by making the best use of your time. There are no long waits while you load editors, compilers, assemblers, and other tools, no long waits while they run—because everything you need is in a single, easy-to-use, 100% resident system. Using polyFORTH, you take a raw idea to fast, compiled code in seconds—and then test it interactively.

polyFORTH has everything you need to develop real-time applications: fast multi-tasking, multi-user OS; FORTH compiler, interpreters, and assemblers; editor and utilities; and over 400 primitives and debugging aids. With its unique modular structure, polyFORTH even helps you test and debug custom hardware interactively, and it is available for most 8, 16, and 32-bit computers.

FORTH, Inc. also provides its customers with such professional support services as custom application programming, polyFORTH programming courses, and the FORTH, Inc. "Hotline."

For more information and a free brochure, contact FORTH, Inc. today. FORTH, Inc., 111 N. Sepulveda Blvd., Manhattan Beach, CA 90266. Phone **(213) 372-8493.**

FORTH, Inc.

Circle **no. 211** on reader service card.

B PROTOCOL

Listing Five (continued from June)

```
            include     \lc\dos.mac

            dseg
Timer                   dw        ?
Old_Second              db        ?
            endds

            pseg

            public      Start_Timer

Start_Timer proc
            push        BP
            mov         BP,SP
            mov         AX,[BP+4]             ; Get the number of seconds
            mov         Timer,AX
            mov         AX,2C00H
            int         21H                            ; Get current time
            mov         Old_Second,DH
            pop         BP
            ret
Start_Timer endp

            public      Timer_Expired

Timer_Expired proc
            mov         AX,2C00H
            int         21H                            ; Get current time
            cmp         DH,Old_Second                  ; Has the clock ticked?
            je          Timer_Expired_1                ; No
            mov         Old_Second,DH                  ; Yes, update Old_Second
            dec         Timer
            cmp         Timer,0
            jle         Timer_Expired_2                ; Timer expired?

Timer_Expired_1:
            xor         AX,AX                          ; No, return "false"
            ret

Timer_Expired_2:
            mov         AX,1                           ; Yes, return "true"
            ret
Timer_Expired endp

            endps
            end
```

End Listing Five

Listing Six

```
#define Loops_Per_Millisecond            9

Delay(N)
/**
 * Delay for N milliseconds
 **/
    int N;
{
    long K;

    for (K = Loops_Per_Millisecond * (long) N; K > 0; K--);
}
```

End Listing Six

Listing Seven

```
        page      57,132
        title     FileIO
;++
; FACILITY: DTE
;
; ABSTRACT:
;
;       This module contains the interface routines to the MS-DOS file
;       service.  All disk operations are done thru this module.
;
; ENVIRONMENT: MS-DOS, V2.0 or later
;
; AUTHOR: Steve Wilhite, CREATION DATE: 8-May-85
;
; REVISION HISTORY:
;
;--
        include   \lc\dos.mac
        pseg
```

```
@ab = 4                                          ; argument base
        page
        public    Create_File

Create_File proc near
;+
; Functional Description:
;
;       Creates a new file or truncates an old to zero length in
;       preparation for writing.
;
; Calling Sequence:
;
;       handle = Create_File(pathname, attribute)
;
; Parameters:
;
;       pathname  ptr to ASCIZ pathname
;       attribute file attribute(s)
;
; Return Value:
;
;       -5        access denied
;       -4        too many open files
;       -3        path not found
;       else      file handle number
;-
Pathname  equ     @ab[BP]
Attribute equ     @ab+2[BP]

        push      BP
        mov       BP,SP
        mov       DX,Pathname
        mov       CX,Attribute
        mov       AH,3CH
        int       21H
        jnc       Create_1
        neg       AX
Create_1:
        pop       BP
        ret
Create_File endp
        page
        public    Open_File

Open_File proc    near
;+
; Functional Description:
;
;       Opens a file.
;
; Calling Sequence:
;
;       handle = Open_File(pathname, access)
;
; Parameters:
;
;       pathname  ptr to ASCIZ pathname
;       access              access code (0 = read, 1 = write, 2 = read and write)
;
; Return Value:
;
;       -12       invalid access
;       -5        access denied
;       -4        too many open files
;       -2        file not found
;       >0        file handle number
;-
Access            equ     @ab+2[BP]

        push      BP
        mov       BP,SP
        mov       AH,3DH
        mov       DX,Pathname
        mov       AL,Access
        int       21H
        jnc       Open_1
        neg       AX
Open_1: pop       BP
        ret
Open_File endp
        page
        public    Close_File

Close_File proc   near
;+
; Functional Description:
;
;       Closes the file associated with a specified file handle.
;
; Calling Sequence:
;
;       status = Close_File(handle)
;
; Parameters:
;
;       handle              file handle for file to close
;
```

(continued on next page)

Vitamin C™
A Healthy Supplement For C Programmers

Perfect Windows, Powerful Data Entry, FULLY Integrated, For Effortless Data Entry Windows!

- Complete input formatting
- Unlimited Validation
- Full attribute control
- Field sensitive application help system
- Multiple virtual windows
- Fully automatic, collision proof overlay and restore
- Print to & scroll background windows
- Animated window "zoom"
- Move, grow, shrink, hide, or show any window
- "Loop function" allows processing while awaiting input

and much much more!

Designed to increase your productivity and help you produce superior applications in dramatically less time! Includes 100% source, tutorial, reference manual, examples, and sample programs.

Specify Microsoft, Lattice, Computer Innovations, Aztec, or Mark Williams. Ask about Unix.

$149.95

NOW... VCScreen !

Our new interactive screen "painter" actually lets you draw your data entry windows! Define fields, text, boxes & borders. Move them around. Change attributes. Then the touch of a button generates C source code calls to the Vitamin C routines! Now, your screen designs will be faster and more pleasing when they are created with VCScreen!

Requires Vitamin C library above. For IBM & compatibles. **$99.95**

For Orders Or Information,

(214)245-6090

Creative Programming Consultants
Box 112097 Carrollton, TX 75011-2097

Include $3 ground, $6 air, $15 overnight shipping, $25 if outside USA. Texans add 6% tax. All funds must be in U.S. dollars drawn on a U.S. bank.

COMBINE THE
RAW POWER OF FORTH
WITH THE CONVENIENCE
OF CONVENTIONAL LANGUAGES

HS/FORTH

Why HS/FORTH? Not for speed alone, although it is twice as fast as other full memory Forths, with near assembly language performance when optimized. Not even because it gives MANY more functions per byte than any other Forth. Not because you can run all DOS commands plus COM and EXE programs from within HS/FORTH. Not because you can single step, trace, decompile & dissassemble. Not for the complete syntax checking 8086/8087/80186 assembler & optimizer. Nor for the fast 9 digit software floating point or lightning 18 digit 8087 math pack. Not for the half megabyte LINEAR address space for quick access arrays. Not for complete music, sound effects & graphics support. Nor the efficient string functions. Not for unrivaled disk flexibility — including traditional Forth screens (sectored or in files) or free format files, all with full screen editors. Not even because I/O is as easy, but far more powerful, than even Basic. Just redirect the character input and/ or output stream anywhere — display, keyboard, printer or com port, file, or even a memory buffer. You could even transfer control of your entire computer to a terminal thousands of miles away with a simple >COM <COM pair. Even though a few of these reasons might be sufficient, the real reason is that we don't avoid the objections to Forth — WE ELIMINATE THEM!

Public domain products may be cheap; but your time isn't. Don't shortchange yourself. Use the best. Use it now!

HS/FORTH, complete system: $395. with "FORTH: A Text & Reference" by Kelly and Spies, Prentice-Hall and "The HS/FORTH Supplement" by Kelly and Callahan

VISA Visa Mastercard MasterCard

HARVARD SOFTWORKS
PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

Circle **no. 132** on reader service card.

B PROTOCOL

Listing Seven (listing continued)

```
; Return Value:
;
;           -6          invalid handle
;           0           no error
;-
Handle                  equ       @ab[BP]

            push        BP
            mov         BP,SP
            mov         AH,3EH
            mov         BX,Handle
            int         21H
            jnc         Close_1
            neg         AX
            jmp         Close_2
Close_1:
            mov         AX,0
Close_2:
            pop         BP
            ret
Close_File endp
            page
            public      Read_File

Read_File proc          near
;+
; Functional Description:
;
;           Transfers a specified number of bytes from a file into a buffer
;           location.  If the returned value for number of bytes read is
;           zero, then the program tried to read from the end of file.
;
; Calling Sequence:
;
;           bytes_read = Read_File(handle, buffer, bytes_to_read)
;
; Parameters:
;
;           handle              file handle for the file to read
;           buffer              ptr to buffer
;           bytes_to_read       number of bytes to read
;
; Return Value:
;
;           -6          invalid handle
;           -5          access denied
;            0          end of file
;           >0          number of bytes actually read
;-
Buffer                  equ       @ab+2[BP]
Count                   equ       @ab+4[BP]

            push        BP
            mov         BP,SP
            mov         AH,3FH
            mov         BX,Handle
            mov         CX,Count
            mov         DX,Buffer
            int         21H
            jnc         Read_1
            neg         AX
Read_1:     pop         BP
            ret
Read_File endp
            page
            public      Write_File

Write_File proc         near
;+
; Functional Description:
;
;           Transfers a specified number of bytes from a buffer into a file.
;           If the number of bytes written is not the same as the number
;           requested, then an error has occurred.
;
; Calling Sequence:
;
;           status = Write_File(handle, buffer, bytes_to_write)
;
; Parameters:
;
;           handle              file handle for file to write
;           buffer              ptr to buffer
;           bytes_to_write      number of bytes to write
;
; Return Value:
;
;           -6          invalid handle
;           -5          access denied
;           else        number of bytes written
;-
            push        BP
            mov         BP,SP
```

```
        mov     AH,40H
        mov     BX,Handle
        mov     CX,Count
        mov     DX,Buffer
        int     21H
        jnc     Write_1
        neg     AX
Write_1:
        pop     BP
        ret
Write_File endp

        public  Move_To_EOF

Move_To_EOF proc
        push    BP
        mov     BP,SP
        mov     AX,4202H
        mov     BX,4[BP]                ; file handle
        xor     CX,CX
        xor     DX,DX
        int     21H
        pop     BP
        ret
Move_To_EOF endp

        endps
        end
```

End Listing Seven

Listing Eight

```
        Title   Serial

        include \lc\dos.mac
        pseg
;+
; Table of Contents:
;-
public  Open_Modem
public  Read_Modem
public  Write_Modem
public  Close_Modem
public  Send_Break

        dseg
Comm_Params equ this byte
        db      11H                     ; XON
        db      13H                     ; XOFF
        db      ?                       ; Baud rate code
        db      0                       ; Parity = none
        db      1                       ; Word length = 8
        db      0                       ; stop bits = 1
        endds

        extrn   AS_Init:near            ; Initialize
        extrn   AS_Set_Mode:near  ; Set XON/XOFF mode
        extrn   AS_Set_Port:near  ; Initialize the port
        extrn   AS_Open:near            ; Open the port
        extrn   AS_IReady:near          ; Test input status
        extrn   AS_IChar:near           ; Input character
        extrn   AS_OReady:near          ; Test output status
        extrn   AS_OChar:near           ; Output character
        extrn   AS_Send_Break:near ; Send a break signal
        extrn   AS_OIdle:near           ; Test output idle status
        extrn   AS_Close:near           ; Close the comm port
        extrn   AS_Term:near            ; Terminate async I/O

;+
; Function:
;       Open the comm port.
;
; Calling Sequence:
;
;       Open_Modem(Port, Rate, Auto_XOFF)
;
; Parameters:
;       Port: 0 = COM1, 1 = COM2
;
;       Rate:   0       110 baud
;               1       300
;               2       450
;               3       1200
;               4       1800
;               5       2400
;               6       4800
;               7       9600
;
;       Auto_XOFF: if true, enable auto XOFF/XON flow-of-control
;-
Open_Modem proc
        push    BP
        mov     BP,SP
        mov     AX,4[BP]                ; Get port number
```

(continued on next page)

Work Smart with These Powerful C Utilities

Get more value from your C system. Boost program quality and slash development time with these professional utilities for leading C-compiler systems.

C Utility Library ~~$185~~ **$155**
Over 300 C subroutines
C and assembler source code and demonstration programs for screen handling, color printing, graphics, DOS disk and file functions, memory management and peripherals control.

C-tree ~~$395~~ **$329**
B-Tree database system
Store, update and retrieve records easily. High-level multi-key ISAM routines and low-level B-Tree functions. Available for MS-DOS, CP/M-86, and CP/M-80. Easily transported. Adaptable for network and multiuser. Includes source.

PHACT ~~$295~~ **$200**
Data Base Record Manager
Includes high-level features found in larger database systems. Available for MS-DOS, CP/M-86 and CP/M-80.

Pre-C ~~$395~~ **$329**
LINT-like source code analyzer
Locates structural and usage errors. Cross-checks multiple files for bad parameter declarations and other interface errors.

Windows for C ~~$195~~ **$165**
Versatile window utility
Supports IBM PC compatible and some non-compatible environments.

PANEL ~~$295~~ **$235**
Screen generating utility
Create custom screens via simple, powerful editing commands. Select colors, sizes and types, edit fields. Includes direct input utility.

HALO ~~$250~~ **$199**
Ultimate C graphics
A comprehensive package of graphics subroutines for C. Supports multiple graphics cards.

PLINK-86 ~~$395~~ **$315**
Overlay linker
Includes linkage editor, overlay management, a library manager and memory mapping. Works with Microsoft and Intel object format.

To order or for information call:

TECWARE
1-800-TEC-WARE
(In NJ call 201-530-6307)

UNIX is a registered TM of Bell Laboratories. C-tree, TM Faircom, Inc., PHACT TM PHACT ASSOC. Pre-C, PLINK-86, TM PHOENIX. HALO TM Media Cybernetics, Inc., PC-lint TM GIMPLE software. PANEL TM Roundhill Computer Systems, Ltd., WINDOWS FOR C TM Creative Solutions. CP/M TM DRI

Circle **no. 109** on reader service card.

C & PASCAL PROGRAMMERS

Blaise Computing provides a broad range of programming tools for Pascal and C programmers, with libraries designed for serious software development. You get carefully crafted code that can be easily modified to grow with your changing needs. Our packages are shipped complete with comprehensive manuals, sample programs and source code.

C TOOLS PLUS

$175.00

NEW! Full spectrum of general-purpose utility functions; windows that can be stacked, removed, and accept user input; interrupt service routines for resident applications; screen handling including EGA 43-line text mode support and direct screen access; string functions; and DOS file handling.

PASCAL TOOLS/TOOLS 2

$175.00

Expanded string and screen handling; graphics routines; easy creation of program interfaces; memory management; general program control; and DOS file support.

VIEW MANAGER

$275.00

Complete screen management; paint data entry screens; screens can be managed by your application program; block mode data entry or field-by-field control. Specify C or IBM/MS-Pascal.

ASYNCH MANAGER

$175.00

Full featured asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; baud rates up to 9600; modem control and XMODEM file transfer. Specify C or IBM/MS-Pascal.

Turbo POWER TOOLS PLUS

$99.95

NEW! Expanded string support; extended screen and window management including EGA support; pop-up menus; memory management; execute any program from within Turbo Pascal; interrupt service routine support allowing you to write memory resident programs; schedulable intervention code.

Turbo ASYNCH PLUS

$99.95

Complete asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; and baud rates up to 9600.

RUNOFF

$49.95

NEW! Text formatter written especially for programmers; flexible printer control; user-defined variables; index generation; and general macro facility. Crafted in Turbo Pascal.

EXEC

$95.00

Program chaining executive. Chain one program from another even if the programs are in different languages. Shared data areas can be specified.

ORDER TOLL-FREE 800-227-8087!

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Circle **no. 159** on reader service card.

B PROTOCOL

Listing Eight (listing continued)

```
         call     AS_Init                          ; Initialize async
         mov      AX,8[BP]              ; Get XOFF/XON flags
         cmp      AX,0                             ;
         je       Init_1
         mov      AX,3
Init_1:
         call     AS_Set_Mode                      ; Set them
         mov      AL,6[BP]              ; Get baud rate code
         mov      Comm_Params[2],AL     ; and store
         mov      SI,offset Comm_Params
         call     AS_Set_Port                      ; Initialize the port
         call     AS_Open                          ; Open the comm port
         pop      BP
         ret
Open_Modem endp

;+
; Function:
;        Read a character from the comm port.
;
; Calling Sequence:
;
;        ret_value = Read_Modem()
;
; Returns:
;        -1 if no character is available; otherwise the character.
;-
Read_Modem proc
         call     AS_IReady             ; Test input status
         cmp      AX,-1                            ; Ready?
         jne      Read_1                           ; Yes
         ret                                       ; No, return -1

Read_1:
         call     AS_IChar              ; Input character
         mov      AH,0
         ret
Read_Modem endp


;+
; Function:
;        Write a character to the comm port.
;
; Calling Sequence:
;
;        status = Write_Modem(Char)
;
; Returns:
;        0 if could not send the character; otherwise -1
;-
Write_Modem proc
         push     BP
         mov      BP,SP
         call     AS_OReady             ; Test output status
         not      AX
         cmp      AX,0                             ; Ready?
         je       Write_1                          ; No, return failure
         mov      AX,[BP+4]             ; Get character to send
         call     AS_OChar              ; Send it
         mov      AX,-1                            ; Success

Write_1:
         pop      BP
         ret
Write_Modem endp


;+
; Function:
;        Close the comm port.
;
; Calling Sequence:
;
;        status = Close_Modem()
;
; Returns:
;-
Close_Modem proc
Close_1:
         call     AS_OIdle              ; Test output idle status
         cmp      AX,0                             ; Done?
         jne      Close_1                          ; No
         call     AS_Close
         call     AS_Term
         ret
Close_Modem endp

;+
; Function:
;        Send a break "character" to the comm port.
```

```
;
; Calling Sequence:
;
;         Send_Break();
;-
Send_Break proc
        mov     AX,50                   ; milliseconds
        call    AS_Send_Break
        ret
Send_Break endp

        endps
        end
```

End Listing Eight

Listing Nine

```
        title   Break
        include \lc\dos.mac
        pseg

        public  Set_Break, Get_Break

Set_Break proc
        push    BP
        mov     BP,SP
        mov     DL,4[BP]        ; Get state to set
        mov     AX,3301H
        int     21H
        pop     BP
        ret
Set_Break endp


Get_Break proc
        mov     AX,3300H
        int     21H
        mov     AL,DL
        xor     AH,AH
        ret
Get_Break endp

        endps
        end
```

End Listings

ST-FORTH $49

For your IBM PC/XT/AT/PCjr

A complete FORTH development system for beginners or experienced users

- 100% FORTH-83 Standard
- ALL source code provided
  Written entirely in FORTH & FORTH assembler
- Powerful, fast, easy-to-use editor
  Search, locate source, edit errors
- Extensive MS-DOS/PC-DOS interface
  Sequential & random files
- Many more useful utilities
- Supplied as MS-DOS .COM file
  Compatible with all versions of DOS
- 148-page User's Manual
- Unconditional money-back guarantee

Beginner's pkg based on Leo Brodie's *Starting FORTH* available

Call or write for brochure and order form.

Sunset Technology
1954 Menalto Ave.
Menlo Park, CA 94025
(415) 325-3680

Circle **no. 183** on reader service card.

ATTENTION

C-PROGRAMMERS

File System Utility Libraries

All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable.

BTree Library 75.00
- High speed **random** and **sequential** access.
- **Multiple keys** per data file with up to **16 million** records per file.
- **Duplicate keys,** variable length data records.

ISAM Driver 40.00
- Greatly speeds application development.
- Combines ease of use of **database manager** with flexibility of **programming language.**
- Supports **multi key files** and **dynamic index definition.**
- Very **easy to use.**

Make 59.00
- Patterned after the **UNIX** utility.
- Works for **programs written in every language.**
- **Full macros, File name expansion** and **built in rules.**

Full Documentation and Example Programs Included.

ALL THREE PRODUCTS FOR – 149.00

For more information call or write:
softfocus
1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

Credit cards accepted.
Dealer inquiries invited.

Circle **no. 259** on reader service card.

Instant-C™: The *Fastest* Interpreter for C

Runs your programs 50 to 500 times faster than any other C language interpreter.

Any C interpreter can save you compile and link time when developing your programs. But only ***Instant-C*** saves your time by running your program at compiled-code speed.

Fastest Development. A program that runs in one second when compiled with an optimizing compiler runs in two or three seconds with ***Instant-C***. Other interpreters will run the same program in two minutes. Or even ten minutes. Don't trade slow compiling and linking for slow testing and debugging. *Only **Instant-C** will let you edit, test, and debug at the fastest possible speeds.*

Fastest Testing. ***Instant-C*** immediately executes any C expression, statement, or function call, and display the results. Learn C, or test your programs faster than ever before.

Fastest Debugging. ***Instant-C*** gives you the best source-level debugger for C. Single-step by source statement, or set any number of conditional breakpoints throughout your program. Errors always show the source statements involved. Once you find the problem, test the correction in seconds.

Fastest Programming. ***Instant-C*** can directly generate executable files, supports full K & R standard C, comes with complete library source, and works under PC-DOS, MS-DOS, or CP/M-86. ***Instant-C*** *gives you working, well-tested programs faster than any other programming tool.* Satisfaction guaranteed, or your money back in first 31 days. ***Instant-C*** is $495.

Rational
Systems, Inc.
P.O. Box 480
Natick, MA 01760
(617) 653-6194

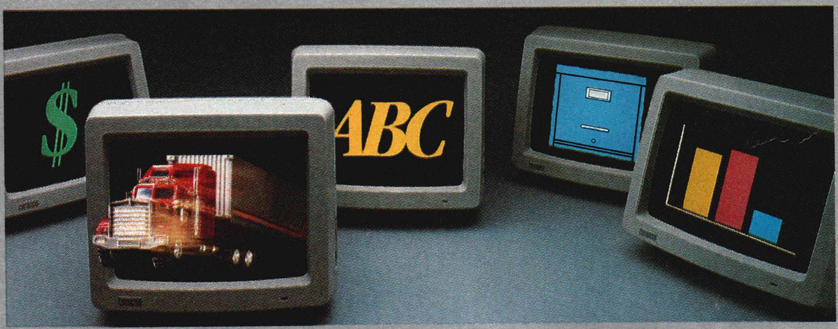
Circle **no. 145** on reader service card.

Listing One (Text begins on page 8.)

```
;Listing 1 - Square root algorithm plus code to benchmark and test it.
;
INT_ROOT  SEGMENT
                ASSUME  CS:INT_ROOT
;
CALC_ROOT PROC     NEAR
;---------------------------------------------;
; Argument passed in BX.                      ;
; Root returned in BX.                        ;
; All registers except BX preserved.          ;
;---------------------------------------------;
                PUSH      AX
                PUSH      CX
                MOV       AX,BX                     ;Hold argument in AX.
                OR        BH,BH
                JNZ       GE_256
                CMP       BL,1                      ;If arg is zero or one,
                JBE       GOT_ROOT ;then root = argument.
                MOV       CL,4                      ;If 2 <= argument <= 255
                SHR       BL,CL                     ;then guess = 3 + arg/16.
                ADD       BL,3
                JMP SHORT NEWTON
GE_256:         MOV       BL,BH
                MOV       BH,0
                JS        GE_32768
                CMP       BL,16
                JAE       GE_4096
                SHL       BL,1                      ;If 256 <= argument <= 4095
                SHL       BL,1                      ;then guess=13+4*(arg hi byte).
                ADD       BL,13
                JMP SHORT NEWTON
GE_4096:  ADD   BL,50                ;If 4096 <= argument <= 32767
                JMP SHORT NEWTON     ;then guess = 50 + arg hi byte
GE_32768: CMP   BL,255       ;If arg hi byte=255 then root=255.
                JZ        GOT_ROOT ;This prevents ovflow by DIV.
                ADD       BL,40                     ;If 32768 <= argument <= 65279
                JNC       NEWTON                    ;then guess = 40 + arg hi byte.
                MOV       BL,255                    ;Guess must never exceed 255.
;
NEWTON:         MOV       CX,AX                     ;Save argument in CX.
                DIV       BL                        ;Divide by guess.
                ADD       BL,AL                     ;Guess + quotient.
                RCR       BL,1                      ;New guess=(old guess+quot)/2.
                MOV       AL,BL                     ;RCR shifts in carry from ADD.;
                MUL       AL                        ;If the square of the new guess
                CMP       AX,CX                     ;is greater than the argument,
                JBE       GOT_ROOT  ;then we decrement new guess
                DEC       BX                        ;to get the correct root.
;
GOT_ROOT: POP   CX
                POP       AX
                RET
;
CALC_ROOT ENDP
;
;Listing 1 - Continued.
;-------------------------------------------------------------;
; Code to time and test CALC_ROOT is designed to be run under;
; DEBUG and does not do a normal return to DOS but instead   ;
; does an INT 3 at the end of each routine.                   ;
;-------------------------------------------------------------;
TIME            PROC      FAR
;-------------------------------------------------------------;
; TIME_ROOT computes the root of each of 65536 possible      ;
; arguments 15 times for a total of 983,040 roots.            ;
; TIME_OVER represents the looping overhead in TIME_ROOT.     ;
; The difference between the two times is the time to call   ;
; and execute CALC_ROOT.                                      ;
;-------------------------------------------------------------;
                MOV       BP,15
TIME_OVER:      MOV       SI,0                      ;Initial value.
INNER_OVR:      MOV       BX,SI
                INC       SI
                JNZ       INNER_OVR
                DEC       BP
                JNZ       TIME_OVER
END_OVER: MOV   AH,2
                MOV       DL,7                      ;Beep speaker.
                INT       21H
                INT       3
;
                MOV       BP,15
TIME_ROOT:      MOV       SI,0                      ;Initial value.
INNR_ROOT:      MOV       BX,SI
                CALL      CALC_ROOT
                INC       SI
                JNZ       INNR_ROOT
                DEC       BP
                JNZ       TIME_ROOT
END_TIME: MOV   AH,2
                MOV       DL,7                      ;Beep speaker.
                INT       21H
                INT       3
;
TIME            ENDP
```


(continued on page 62)

Digital has it now.



The first open-ended multi-user software system that integrates your custom application with a full range of standard business programs.

The A-to-Z™ Integrated System for MicroVAX II™ and MicroPDP-11.™ Now you can integrate any number of your custom multi-user applications with word processing, graphics, spreadsheet, data management, accounting, and thousands of VAX™ and PDP-11 programs. Same menus, commands and files with no need for a system manager.

Tell me about A-to-Z on MicroVAX™ and MicroPDP-11.™ ☐ Reseller ☐ Developer

Name ____________________ Title ____________________

Company ____________________ Phone ____________________

Address ____

City ____________________ State __________ Zip __________

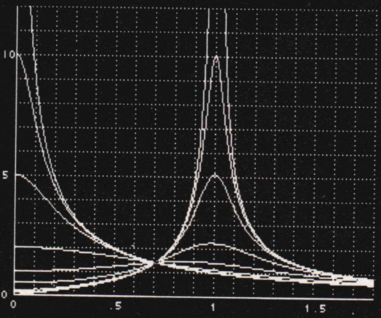
Send to: Digital Equipment Corporation, Inquiry Dept., NR02-1/H3, 444 Whitney Street, Northboro, MA 01532.

digital™

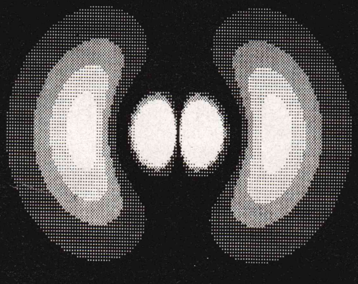
© Digital Equipment Corporation 1986. Digital, the Digital logo, A-to-Z, MicroVAX II, MicroPDP-11, and VAX are trademarks of Digital Equipment Corporation. DD786

isys forth

for the Apple®][



Parallel Resonance with Damping
BASIC 213 sec   ISYS FORTH 20 sec



Hydrogen 3p Orbital Cross-section
BASIC 492 sec   ISYS FORTH 39 sec

ISYS FORTH is a FORTH-83 compiler designed especially for scientific and engineering applications.

- **FASTEST** — Sieve benchmark, 3.3 seconds/pass. Compiles to machine language.
- **FLOATING POINT WITH TRANSCENDENTALS,** single and double precision.
- **16-BIT 65802 SUPPORT** for execution speeds 50-100% faster than the above.
- **FIXED POINT THEORY AND EXAMPLES** (a 30-page chapter). D*/ for double precision fixed point.
- **TURTLE AND CARTESIAN GRAPHICS** with 70-column character set and double hires support.
- **MACRO ASSEMBLER.**
- **DOS 3.3 FILE INTERFACE.**
- **FULL-SCREEN EDITOR.**
- **144-PAGE MANUAL WITH TUTORIAL.**
- **PRICE: $99, NO EXTRA CHARGES.**

ILLYES SYSTEMS
PO Box 2516, Sta A
Champaign, IL 61820
Phone: 217/359-6039

For any Apple][model, 48K or larger.
Apple is a registered trademark of Apple Computer.

Circle **no. 99** on reader service card.

Listing One (Listing continued, text begins on page 8.)

```
;
;Listing 1 - Continued.
;
TEST                PROC      FAR
;-----------------------------------------------------------;
; If CHK_ROOT detects a bad root, it displays a message and  ;
; leaves the NG root in BX and the argument in SI.           ;
; If all roots are OK, a message to this effect is displayed.;
;-----------------------------------------------------------;
                    MOV       SI,0            ;Initial value.
CHK_ROOT: MOV       BX,SI
                    CALL      CALC_ROOT
                    MOV       AX,BX
                    MUL       AX              ;DX,AX contains root^2.
                    JO        NG_ROOT         ;NG if root ^2 > 65535.
                    CMP       AX,SI
                    JA        NG_ROOT         ;NG if root^2 > argument.
                    MOV       AX,BX
                    INC       AX
                    MUL       AX              ;DX,AX contains (root+1)^2.
                    JO        NEXT_ARG ;If ovflow then OK.
                    CMP       AX,SI
                    JBE       NG_ROOT         ;NG if (root+1)^2 <=argument.
NEXT_ARG: INC       SI
                    JNZ       CHK_ROOT
                    JMP SHORT OK_ROOTS
;
OK_MSG    DB  0DH,0AH,'All roots tested OK.',0DH,0AH,'$'
NG_MSG  DB  0DH,0AH,'Bad root in BX. Arg in SI.',0DH,0AH,'$'
;
OK_ROOTS; MOV       DX,OFFSET OK_MSG+100H     ;DS points to Pgm Seg
                    JMP SHORT DO_MSG          ;Pref which is 100H
NG_ROOT:  MOV       DX,OFFSET NG_MSG+100H     ;lower than code seg.
;
DO_MSG:             MOV       AH,9            ;Print string.
                    INT       21H
                    INT 3                     ;Back to DEBUG.
;
TEST                ENDP
;
INT_ROOT  ENDS
;
                    END       TEST
```

End Listing One

Listing Two

```
;Listing 2 - BASIC program to test if a formula makes good square root guesses.

10 FOR I = 2 TO 256
20 Q = I*I - 1
30 '
40 'Trial Formula to Calculate P0.
50 '
60 QHI = INT(Q/256): QLO = Q-QHI*256
70 IF QHI = 0 THEN P0 = INT(QLO/32) + 3: GOTO 160
80 IF QHI < 16 THEN P0 = 13 + 4*QHI: GOTO 160
90 IF QHI < 128 THEN P0 = QHI + 50; GOTO 160
100 IF QHI = 255 THEN P1 = 255: GOTO 210
110 P0 = QHI + 40
120 IF P0 > 255 THEN P0 = 255
130 '
140 ' Newtons Method
150 '
160 P1=INT((P0 + INT(Q / P0)) / 2)
170 IF P1 > 255 THEN PRINT "P1 > 255 when Q = ";Q:END
180 '
190 'Test result
200 '
210 P = INT(SQR(Q))
220 IF P1 <= P+1 GOTO 240
230 PRINT "For Q = ";Q;" P1 is greater than P+1. ":END
240 NEXT I
250 PRINT "Formula works for all worst cases."
```

End Listing Two

Listing Three

```
Listing Three

 INCLUDE MACLIB.ASM          ;by Neil R. Koozer
 LIST ON                     ;   Kellogg Star Rt. Box 125
 MACLIST OFF                         ;   Oakland, OR 97462
;                                         (503)-459-3709
;SQR.ASM

;Note that words like BR1 and BFS1 are macros to emulate BR:B and BFS:B

GLOBAL SQR

SQR                   ;square root function for 32000 floating point
 RESTORE [R0]             ;use ret. addr as a pointer
 MOVD 0(R0),R1        ;get operand address
 MOVD 0(R1),R6        ;get part of operand
 MOVD 4(R1),R7        ;get other part of operand
 SBITB 31,R7          ;make the implicit 1 explicit
```


(continued on page 66)

Your personal connection
for artificial intelligence software

Fundamentally new ideas don't come along very often, but when they do, they can be very rewarding. Artificial Intelligence and Expert Systems are emerging technologies that promise to revolutionize the software industry. We at Programmer's Connection are pleased to be your personal connection for these and other fine products for IBM Personal Computers and compatibles.

EXPERT EDGE by Human Edge 795 **659**

Powerful expert system builder for creating interactive knowledge-based programs. It has a natural language interface that makes creating an expert system advisor easy by prompting for rule-based knowledge one step at a time. There's also a special user language that allows you to use your text editor to create or modify knowledge bases. Knowledge checking examines your knowledge bases for redundancies and conflicts. It supports Baysian statistics and probabilities and provides full control over the screen, windows, color, data formats, numeric punctuation and much more. DIF and SYLK formats are supported for easy interface to other programs' data. Resulting systems use an interactive dialog to communicate with the user providing fast responses, reasoning justification, context sensitive helps, printed reports, footnotes and much more. Requires 256K memory (512K recommended for large programs of up to 500 rules).

Experteach II by Intelligenceware 475 **389**

Comprehensive guide to expert system concepts and technology that provides a variety of AI tools for expert system experimentation. An on-line tutorial describes the operation of expert systems. It includes both a forward chaining and a backward chaining expert system shell implemented in each of LISP, Prolog, Pascal and dBase II. They support inexact reasoning with variables and how & why explanations. Expert system shells may be edited with a built-in rule editor. Includes source code and design documents for all eight expert system shells. The system comes with both a complete Lisp interpreter and a complete Prolog interpreter. It also includes comprehensive case studies of several major expert systems and a comprehensive bibliography. Requires 256K memory.

EXSYS by EXSYS Inc. 475 **339**
Runtime License 600 **479**

Development software for creating IF-THEN-ELSE probabilistic knowledge based expert systems. It can interface with other programs and includes a built-in report generator to print expert system analyses in report format. It includes an editor program, a copy-protected runtime program, an expert system file compression utility, a utility to arrange rules for maximum speed, demonstration programs and more. The developer's Runtime License includes a copy of the runtime program without copy-protection and permission to copy and distribute an unlimited number of copies of the runtime program with no royalties. Requires 256K and uses full available memory.

GCLISP Golden Common Lisp by Gold Hill Computers
GCLISP 495 **CALL**
GCLISP 286 Developer 1190 **CALL**

Artificial intelligence development language that is a major subset of COMMON Lisp. GCLISP includes a small model interpreter, the GMACS editor, the San Marco LISP Explorer, on-line help facilities, the book LISP by Winston & Horn, the COMMON LISP Reference Manual by Guy Steele and the GCLISP user manual. GCLISP 286 Developer also includes a large model interpreter that can access up to 15 megabytes of memory and a large model compiler that allows applications to run up to 15 times faster. Compiled code may call interpreted code and vice versa. GCLISP is copy protected and requires an IBM PC or AT or 100% compatible with at least 512K memory. GCLISP 286 Developer is not copy protected and requires an IBM AT or 100% AT compatible with at least 512K memory.

INSIGHT by Level 5 Research
INSIGHT 1 50 **45**
INSIGHT 2+ 485 **389**

Powerful expert system development tools that allow you to quickly and easily develop knowledge-based systems. Knowledge bases are written in the English-like Production Rule Language (PRL) and then streamlined into compiled code that is run by INSIGHT. This series of software is upwardly compatible to allow plenty of room for growth. INSIGHT 1 is an expert system primer designed for small systems, prototyping and job aids. It combines IF-THEN-ELSE rules and goal outlining to define knowledge that is processed with classical backward chaining reasoning. INSIGHT 2+ includes a Pascal compiler, Pascal interpreter, a dBase interface, a Turbo Pascal interface, intrinsic math functions, advanced knowledge representation and a backward and forward chaining/goal driven inference engine. Requires 512K memory.

Microsoft LISP 250 **189**

Advanced implementation of the LISP language that supports over 400 Common LISP commands, macros, special forms and control variables. It provides extremely fast performance and can be used to create exceptionally compact programs and data files. The system takes full advantage of up to 512K memory and features an efficient, two-pass garbage collector that performs automatic, dynamic-memory management over all data spaces. The package includes a powerful, symbolic LISP debugger that features a multi-level break-and-trace facility and an execution profiler. There are separate windows for editing and debugging. Comes with an interactive tutorial.

Q'NIAL by NIAL Systems 375 **359**

Nested Interactive Array Language is a very high level fifth generation programming language that combines ideas from APL, LISP and structural and functional programming. It provides the best features of present day languages in an environment suitable for rapid programming. This highly extensible language provides symbolic data handling that makes words, letters and sentences as easy to handle as numeric data. All data elements are defined as nested rectangular arrays. Data structures are displayed with boxes that clearly indicate the content and structure of arrays. Applications are designed in immediate mode and may be run in either batch or immediate mode. Includes a host interface allowing execution of DOS programs. It comes with a system editor, library of programs and graphics support.

Turbo Prolog by Borland International 100 **79**

Complete Prolog development system that includes a compiler, an interactive editor, a 200 page reference manual with a step-by-step tutorial and demo programs. This very fast incremental compiler compiles over 2500 lines per minute on a standard IBM PC. It generates native in-line code and object modules that are linkable with the standard DOS linker. When the compiler encounters an error, the interactive full screen text editor is invoked with the cursor placed over the offending code. It supports a flexible, object-oriented type system, graphic windowing, text windowing, and full I/O facilities including formatted I/O, streams and random access files. Integer values range from −32768 to 32767; real values range from 1E-307 to 1E+308. Complete built-in trace debugging facilities allow single stepping of programs and the ability to view the running program's source code. Includes Geobase, a free natural query language database designed and developed around U.S. geography that includes commented source code on disk. Requires 384K memory.

CALL TOLL FREE: U.S. 800-336-1166
CANADA 800-225-1166 OH 216-877-3781

programmer's connection

Turn the page for a wide selection of programmer's development tools exclusively for IBM Personal Computers and compatibles.

Circle **no. 129** on reader service card.

PROGRAMMER DEVELOPMENT TOOLS FOR THE IBM-PC/XT/AT
and compatibles.

NO
*Shipping Charge**
Handling Charge
Insurance Charge
Credit Card Charge
C.O.D. Charge
Purchase Order Charge
Hidden Charges
PLUS OUR NO RISK GUARANTEE

*When shipping via standard United Parcel Service.

apl language

APL*PLUS/PC System by STSC	595	**449**
APL*PLUS/PC Tools Vol 1 by STSC	295	**239**
APL*PLUS/PC Tools Vol 2 by STSC	85	**69**
APL*PLUS Spreadsheet Manager by STSC ... *New*	195	**159**
APL*PLUS/UNX System *For AT Xenix* by STSC	995	**795**
Btrieve *ISAM File Mgr with No Royalties* by SoftCraft	250	**195**
Financial/Statistical Library by STSC	275	**219**
FRESCO *Business Graphics Library* by Mr. APL	300	**269**
Pocket APL by STSC	95	**79**
STATGRAPHICS *Statistical Graphics System* by STSC	695	**539**

artificial intelligence

ExpertEASE by Human Edge	695	**589**
ExpertEDGE by Human Edge	795	**659**
Experteach *Complete System* by Intelliware	475	**389**
EXSYS *Expert System Development Software* by EXSYS	395	**339**
GCLISP *Golden Common LISP* by Gold Hill	495	**CALL**
GCLISP 286 Developer *LM Interpreter & LM Compiler*	1190	**CALL**
Insight 1 *AI Primer* by Level Five Research	50	**45**
Insight 2+ by Level Five Research ... *New*	485	**389**
Microsoft LISP *Common LISP*	250	**189**
Methods *Smalltalk-based Prototyping* by Digitalk	250	**209**
PLA microProlog by Programming Logic Associates	250	**219**
with APES	450	**399**
PLA Professional microProlog by Programming Logic	395	**349**
with APES	695	**599**
Prolog-86 from Solution Systems	95	**CALL**
Prolog-86 Plus from Solution Systems	250	**CALL**
QNIAL *Combines APL with LISP* by NIAL Systems	375	**359**
Small-X by Kaplan	125	**99**
TransLISP from Solution Systems	75	**CALL**
Turbo Prolog Compiler by Borland International	100	**79**

assemblers and debuggers

8088 Assembler *w/Z-80 Translator* by 2500 AD	100	**89**
Advanced Trace-86 *with ASM Interpreter* by Morgan	175	**139**
Codesmith-86 Debugger by Visual Age	145	**109**
Cross Assemblers *Over 25 Varieties* from 2500 AD	CALL	**CALL**
Microsoft Macro Assembler *with utilities*	150	**99**
Periscope I *w/Board & Switch* by Data Base Decisions	295	**249**
Periscope II *w/NMI Breakout Switch Only*	145	**115**
Periscope II-X *Software only* ... *New*	95	**85**
The PROFILER *with Source Code* by DWB Associates	125	**95**
Turbo EDITASM *Fast Assembler* by Speedware	99	**84**
Visible Computer: 8088 by Software Masters	80	**65**

basic language

BetterBASIC by Summit Software	200	**165**
8087 Math Support	99	**85**
Btrieve Interface	99	**85**
C Interface ... *New*	CALL	**CALL**
Run-time Module	250	**225**
Microsoft QuickBASIC Compiler *New version*	99	**79**
Professional BASIC by Morgan Computing	99	**79**
8087 Math Support	50	**47**
True Basic from Addison-Wesley	150	**105**
Run-time Module	500	**435**

blaise products

Asynch Manager *Specify for C or Pascal*	175	**139**
C Tools Combination Package *Both Items Below*	175	**149**
C Tools	125	**105**
C Tools 2	100	**84**
Exec *Program Chainer*	95	**79**
Pascal Tools Combination Package *Both Items Below*	175	**149**
Pascal Tools	125	**105**
Pascal Tools 2	100	**84**
Turbo ASYNCH *for Turbo Pascal*	100	**84**
Turbo POWER TOOLS *for Turbo Pascal*	100	**84**
View Manager *Specify for C or Pascal*	275	**209**
with Source Code	295	**239**

borland products

REFLEX *Data Base System*	99	**75**
Turbo DATABASE TOOLBOX	55	**38**
Turbo EDITOR TOOLBOX	70	**54**
Turbo GAMEWORKS TOOLBOX	70	**54**
Turbo GRAPHIX TOOLBOX	55	**38**
Turbo LIGHTNING	99	**75**
Turbo PASCAL	70	**49**
with 8087 or BCD	110	**77**
with 8087 and BCD	125	**84**
Turbo Prolog Compiler	100	**79**
Turbo TUTOR *for Turbo PASCAL*	35	**28**

c compilers

C-86 Compiler ... *See Computer Innovations Section*	395	**289**
Datalight C Compiler *Small Memory Model*	60	**49**
Datalight Developer's Kit *with Large Memory Model*	99	**79**
DeSmet C Compiler *with Source Debugger*	159	**145**
Eco-C *Complete Development System* by Ecosoft	125	**89**
Lattice C Compiler ... *See Lattice Section*	500	**299**
Let's C Compiler by Mark Williams	75	**69**
with csd Source Level Debugger	150	**129**
Microsoft C Compiler	395	**259**
MWC-86 by Mark Williams ... *Special Price*	495	**289**
Wizard C Compiler *Includes Lint* by Wizard Systems	450	**369**

c interpreters

C-terp by Gimpel Software *Specify compiler interface*	300	**239**
Instant C by Rational Systems	500	**379**
Interactive C by IMPACC Associates	249	**219**
Run/C by Age of Reason	150	**99**
Run/C Professional by Age of Reason	250	**189**

c utilities

Also refer to Blaise, Computer Innovations, Lattice, Microsoft, Phoenix, Polytron, SoftCraft and Xenix System V sections.

APT Application Programmer's Toolkit by Shaw American	395	**339**
Basic C Library by C Source	175	**135**
C Essentials by Essential Software	100	**85**
C-lib by vance info systems	195	**125**
C Power Packs by Software Horizons	CALL	**CALL**
c-tree *ISAM File Manager with source* by FairCom	395	**329**
C Utility Library by Essential Software	185	**139**
C Windows by Syscom	100	**89**
C Wings by Syscom	50	**45**
dbVISTA *Single-User DBMS* by Raima	195	**159**
with Source Code	495	**429**
dbVISTA *Multi-User DBMS* by Raima	495	**429**
with Source Code	990	**849**
EditCheck by Everest Solutions	90	**79**
Entelekon Combo Package *All 3 items below*	200	**175**
C Function Library	130	**115**
C Windows	130	**115**
Superfonts for C	50	**45**
Essential Graphics *No Royalties* by Essential Software	250	**219**
Flash-up Windows by Software Bottling of NY	75	**69**
GraphiC *Mono version 2.2* by Scientific Endeavors	280	**219**
GraphiC *Color version 3.0* by Scientific Endeavors	350	**299**
The Greenleaf Functions by Greenleaf Software	185	**135**
Greenleaf Comm Library by Greenleaf Software	185	**135**
The HAMMER by OES Systems	195	**175**
MetaWINDOWS by Metagraphics	185	**139**
MetaWINDOWS/Plus by Metagraphics	235	**199**
Multi-Halo *with Royalties* by Media Cybernetics	300	**219**
On-line Help from Opt-Tech Data Processing	149	**119**
PANEL by Roundhill *Library Source Also Available*	295	**229**
PC Lint by Gimpel Software	139	**109**
Scientific Subroutine Library for C by Peerless	175	**139**
Vitamin C by Creative Programming	150	**139**
VC Screen *Interactive Forms Designer*	99	**85**
Zview by Data Management Consultants	245	**199**

cobol language

Micro Focus COBOL Workbench	4000	**3599**
Micro Focus Level II COBOL ... *Special Price*	1500	**CALL**
COMATH	200	**169**
FORMS-2	300	**269**
Level II Animator ... *Special Price*	900	**CALL**
Level II SOURCEWRITER ... *Special Price*	2000	**CALL**
Micro Focus Level II COBOL *For Novell NetWare* ... *New*	2000	**1799**
Micro Focus Micro/SPF	175	**159**
Micro Focus Professional COBOL	3000	**2395**
Multi-user Runtime *for PC Network* ... *New*	500	**449**
Microsoft COBOL ... *See Microsoft Section*		
OPT-Tech Sort *Also Sorts Btrieve Files*	149	**119**
Realia COBOL	995	**795**
RM/COBOL by Ryan-McFarland	950	**675**
RM/COBOL 8X *ANSI 85 COBOL* by Ryan-McFarland	1250	**995**

computer innovations products

C-86 Optimizing Compiler	395	**289**
C to dBase	150	**139**
CI Probe *Source Level Debugger*	225	**199**
CI RomPac *for C-86*	195	**149**
Introducing C *C Interpreter*	125	**105**

forth language

CFORTH *Native Code Application Compiler* by LMI ... *New*	300	**239**
LMI Forth/83 Metacompiler *Specify Target Processor* ... *New*	750	**599**
PC/Forth by Laboratory Microsystems	150	**119**
PC/Forth+ by Laboratory Microsystems	250	**209**
Advanced Color Graphics Support ... *New*	100	**79**
Enhanced Graphics Support ... *New*	200	**159**
Intel 8087 Support ... *New*	100	**79**
Interactive Symbolic Debugger ... *New*	100	**79**
Native Code Optimizer ... *New*	200	**159**
PCTERM *Modem Program for Smartmodem* ... *New*	100	**79**
Software Floating Point ... *New*	100	**79**

fortran language

ACS Time Series by Alpha Computer Service	495	**429**
Btrieve *ISAM File Manager* ... *See SoftCraft Section*		
For-Winds by Alpha Computer Service	90	**79**

Circle **no. 129** on reader service card.

fortran language *continued*

Forlib-Plus by Alpha Computer Service	70	55
Microsoft Fortran	350	209
MORE FORTRAN by Peerless Engineering	125	99
Multi-Halo *with Royalties* by Media Cybernetics	300	219
PANEL *Screen Designer* by Roundhill	295	229
PC Fortran Tools by Stat Com Systems	179	159
PolyFortran Tools by Polytron	179	139
RM/Fortran by Ryan-McFarland	595	395
Scientific Subroutine Library by Peerless	175	139
The Statistician by Alpha Computer Service	295	259
Strings & Things by Alpha Computer Service	70	55

lattice products

Lattice C Compiler	500	299
with Library Source Code	900	549
C Cross Reference Generator	50	39
with Source Code	200	159
C-Food Smorgasbord *Function Library*	150	99
with Source Code	300	195
C-Sprite *Source Level Debugger*	175	139
Curses *Screen Manager*	125	99
with Source Code	250	199
dBC *dBase File Manager for C*	250	199
with Source Code	500	395
LMK *Make Facility*	195	149
LSE *Screen Editor*	125	99
RPG II Compiler *No Royalties*	750	595
SecretDisk *File Security*	60	49
SideTalk *Resident Communications*	120	95
Text Mgmt Utilities *(GREP,DIFF,ED,WC,Extract,Build)*	120	95
TopView Toolbasket *Function Library*	250	199
with Source Code	500	395
Z-80 C Cross Compiler	500	395
with Library Source Code	1000	789

microsoft products

Microsoft BASIC Interpreter for Xenix	350	279
Microsoft C Compiler	395	259
Microsoft COBOL Compiler	700	495
for Xenix	995	795
Microsoft COBOL Tools *with COBOL Source Debugger*	350	209
for Xenix	450	359
Microsoft Fortran Compiler	350	209
for Xenix	495	389
Microsoft LISP *Common LISP*	250	189
Microsoft Macro Assembler *with utilities*	150	99
Microsoft Mouse *Bus Version*	175	149
Microsoft Mouse *Serial Version*	195	159
Microsoft muMath *Includes muSIMP*	300	195
Microsoft Pascal Compiler	300	195
for Xenix	495	389
Microsoft QuickBASIC Compiler *New version*	99	79
Microsoft Sort	195	149
Microsoft Windows	99	74
Microsoft Windows Developer's Kit *New*	500	CALL

other products

Dan Bricklin's Demo Program by Software Garden	75	65
FASTBACK *Backup Utility* by 5th Generation Systems	179	159
Interactive EASYFLOW by Haventree Software	150	129
Janus/ADA C Pack by R&R Software	95	89
Janus/ADA D Pack by R&R Software	900	699
MODULA-2/86 by Logitech	CALL	CALL
SET:SCIL by System Engineering Tools *New*	349	299
Source Print by Aldebaran Laboratories *New*	97	89
SRMS *Software Revision Mgmt System* by Quilt *New*	125	109

phoenix products

Authorized Dealer

Pasm86 *Macro Assembler*	295	195
Pfantasy Pac *(Pfinish,Pfix+,Plink+,Pmaker,Pmate,Ptel)*	1295	949
Pfinish *Performance Analyzer*	395	249
Pfix-86 Plus *Symbolic Debugger*	395	249
PforCe *Comprehensive C Function Library*	475	319
Plink-86 *Overlay Linker*	395	249
Plink-86 Plus *Enhanced Overlay Linker*	495	389
Pmaker *Program Development Manager*	195	139
Pmate *Macro Text Editor*	225	149
Pre-C *Lint Utility*	395	249
Ptel *Binary File Transfer Program*	195	139

polytron products

Polytron C Beautifier	49	45
Polytron C Library I	99	79
Polytron PowerCom *Communications*	179	139
PolyFORTRAN Tools I	179	139
PolyLibrarian *Library Manager*	99	79
PolyLibrarian II *Library Manager*	149	119
PolyMake *UNIX-like Make Facility*	99	79
PolyOverlay *Overlay Optimizer*	99	79
PolyWindows Developer Kit	199	149
PolyWindows Products *All Varieties*	CALL	CALL
PolyXREF *Complete Cross Reference Utility*	219	179
PolyXREF *Support for one language only*	129	109
PVCS *Polytron Version Control System*	395	329
PVMFM *Polytron Virtual Memory File Manager*	199	149

softcraft products

Btrieve *ISAM File Manager with No Royalties*	250	195
Xtrieve *Query Utility for Btrieve*	195	169
Rtrieve *Report Generator for Xtrieve*	85	79
Btrieve/N *for Networks*	595	465
Xtrieve/N *Query Utility for Btrieve/N*	395	299
Rtrieve/N *Report Generator for Xtrieve/N*	175	159

text editors

Brief from Solution Systems	195	CALL
Epsilon *Multi-tasking Emacs-like editor* by Lugaru	195	165
FirsTime for Turbo by Spruce Technology	75	69
KEDIT *Xedit-like editor* by Mansfield Software Group	125	109
SPF/PC by Command Technology Corp	195	165
Vedit by CompuView	150	115
Vedit Plus by CompuView	225	180
XTC *Text Editor with source* by Wendin	99	84

turbo pascal utilities

Also refer to Blaise, Borland and SoftCraft sections.

ALICE *Turbo Pascal Interpreter* by Software Channels	95	85
FirsTime for Turbo by Spruce Technology	75	69
Flash-up Windows by Software Bottling of NY	75	69
Multi-Halo *with Royalties* by Media Cybernetics	300	CALL
On-line Help from Opt-Tech Data Processing	149	119
Screen Sculptor by Software Bottling of NY	125	95
Turbo EXTENDER by TurboPower Software	85	69
Turbo Professional by Sunny Hill Software	70	49
TurboPower Utilities by TurboPower Software	95	84
TurboRef by Gracon Services	50	45
TurboWINDOW by MetaGraphics	80	69

video training tapes

These video cassette training tapes are from The Information Factory and are an excellent alternative to expensive classroom training. Specify Beta of VHS. Price includes one student manual.

Basic Telecommunications *New*	350	309
Computer Literacy	400	349
Local Area Networks	350	309
Programmer's Introduction to C	500	449

wendin products

Operating System Toolbox *Build your own OS*	99	84
PCUNIX *Operating System*	99	84
PCVMS *Operating System Similar to VAX/VMS*	99	84
XTC *Text Editor with Pascal Source Code*	99	84

xenix system v

Complete Xenix System by SCO *All 3 items below*	1295	1099
Xenix Development System	595	529
Xenix Operating System *Specify XT or AT*	595	529
Xenix Text Processing Package	195	179

xenix languages and utilities

APL*PLUS/UNX System *For AT Xenix* by STSC	995	795
Btrieve *ISAM File Manager* by SoftCraft	595	465
c-tree *ISAM File Manager with Source* by FairCom	395	329
dbVISTA *Single and Multi User versions* by Raima *New*	CALL	CALL
Informix by RDS	995	839
Lyrix by SCO	595	489
Micro Focus Level II Compact COBOL *For AT* *New*	1000	899
Forms-2 *New*	400	359
Level II ANIMATOR *New*	600	539
Microsoft Languages *See Microsoft Section*		
Networks for Xenix by SCO	595	529
PANEL *Screen Designer for AT Xenix by Roundhill*	595	539
RM/COBOL by Ryan-McFarland *New*	1250	995
RM/FORTRAN by Ryan-McFarland *New*	750	599
SCO Professional *Complete Lotus clone* by SCO	795	695

Prices are subject to change without notice.

We are open until 5 p.m. Pacific Time, (8 p.m. Eastern). Visa and MasterCard are accepted with no surcharge applied. Please include card expiration date when ordering by mail. Account is charged when shipped.

VISA MasterCard

CALL TOLL FREE

800·336·1166
U.S. OHIO 216-877-3781

800·225·1166
CANADA

OUR NO RISK GUARANTEE

If you are not completely satisfied with your purchase you may return it within 30 days. All returned products must meet our standards for being in new, resellable condition including all paperwork and unused registration card. Products including source code are generally excluded by the manufacturer from this guarantee. Please ask for specific details when placing your order.

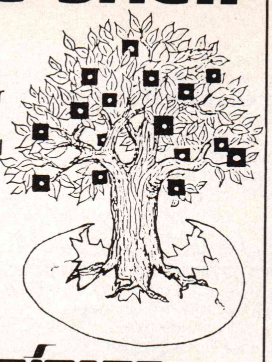
programmer's connection

136 SUNNYSIDE ST.
HARTVILLE, OHIO 44632

Circle **no. 129** on reader service card.

Tree Shell

A Graphic Visual Shell for Unix/Xenix End-Users and Experts Alike!



Dealer inquiries welcomed.

COGITATE

"A Higher Form of Software"
24000 Telegraph Road
Southfield, MI 48034
(313) 352-2345
TELEX: 386581 COGITATE USA

Circle **no. 81** on reader service card.

C Users' Group

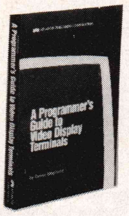
Over 85 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

The C Users' Group
Post Office Box 97
McPherson, KS 67460
(316) 241-1065

Circle **no. 181** on reader service card.



PROGRAMMERS: Now your software can support over 150 video display terminals.

If you write software you need this book. It contains a tutorial on programming for video display terminals plus data sheets to support over 150 VDTs, including **cursor positioning, clear screen, clear-to-end of line, row and column numbering, video attributes, function keys and much more!** You'll find detailed information needed to support a wide range of VDTs immediately, getting your programs to market much earlier.

"Highly recommended for programmers."
— Jerry Pournelle, BYTE Magazine

335 pages, 6" × 9", softcover, ISBN 0-936158-01-8
15-day money-back guarantee. Texas residents add sales tax. Foreign residents specify and add shipping. Send $30 check, MC or VISA, to:

Atlantis Publishing Corporation
Dept. 208, POB 59467, Dallas, TX 75229

Circle **no. 241** on reader service card.

Listing Three **(Listing continued, text begins on page 8.)**

```
 MOVD R6,R1           ;get exponent
 ANDW 7FFH,R1         ;clean exponent
 ADDW 3FFH,R1         ;fix offset
 ROTW -1,R1           ;div exp by 2
 CBITB 15,R1          ;test & clear wrap-around bit, 1=odd 0=even
 SAVE [R0,R1]         ;save exp & ret addr
 MOVB R7,R6           ;prepare for right shift
 BFS1 SQR1            ;jump if exponent was odd
 LSHD -4,R7           ;shift -3 for safety & -1 to get halfx
 ROTD -4,R6
 ANDW FF80H,R6        ;remove non-mantissa bits
 MOVD 4C1BF828H,R3    ;y seed = 1.189.../2
 BR1 SQR2
SQR1
 LSHD -3,R7           ;shift -3 for safety, -1 to get halfx, +1 because
 ROTD -3,R6           ;  orig exp was odd
 ANDW FF00H,R6        ;remove non-mantissa bits
 MOVD 6A227E65H,R3    ;y seed = 1.68.../2

SQR2                  ;We will do 3 iterations with 32-bit precision
 MOVD R7,R5           ;get halfx into R5
 DEID R3,R4           ;R5 = halfx/y0  (the junk in R4 doesn't matter)
 LSHD -1,R3           ;R3 = y0/2
 ADDD R5,R3           ;R3 = new y0

 MOVD R7,R5           ;second iteration
 DEID R3,R4
 LSHD -1,R3
 ADDD R5,R3

 MOVD R7,R5           ;third iteration
 DEID R3,R4
 LSHD -1,R3
 ADDD R5,R3


 MOVD R6,R4           ;Now the final iteration at full precision
 MOVD R7,R5           ;get R4R5 = halfx from R6R7

 DEID R3,R4           ;now divide halfx (R4R5) by y (R2R3)
 MOVD R2,R0
 MEID R5,R0
 NEGD R0,R0
 SUBCD R1,R4
 MOVD R4,R1
 BCC1 DIV1
DIV2
 ADDQD -1,R5
 ADDD R2,R0
 ADDCD R3,R1
 BCC DIV2
DIV1
 DEID R3,R0
 MOVD R1,R4           ;R4R5 now = halfx / y

 MOVB R3,R2
 LSHD -1,R3
 ROTD -1,R2           ;R2R3 = y/2

 ADDD R4,R2
 ADDCD R5,R3          ;R2R3 = y/2 + halfx/y
                      ;4th iteration complete

 RESTORE [R0,R1]      ;get exponent & ret. addr
 ADDD R2,R2           ;shift mantissa back where it belongs
 ADDCD R3,R3
 BCC1 SQR4            ;there should almost never be a carry
 MOVB R3,R2
 LSHD -1,R3           ;undo that last shift & zero the MSbit
 ROTD -1,R2
 ADDQW 1,R1           ;adjust exponent
 BR1 SQR5             ;done
SQR4
 CBITB 31,R3          ;test & clear MSbit (make it a + sign)
 BFS1 SQR5            ;it would virtually always be a 1
 ADDD R2,R2           ;if not, shift left again
 ADDCD R3,R3
 ADDQW -1,R1          ;adjust exponent
 CBITB 31,R3          ;make it +
SQR5
 ANDW F800H,R2        ;clean the mantissa
 ORW R1,R2            ;append the exponent

 MOVD 4(R0),R1        ;get addr of destination variable
 MOVD R2,0(R1)        ;store result in dest. variable
 MOVD R3,4(R1)
 JUMP 8(R0)           ;return to caller

 END
```

End Listings

THE PROGRAMMER'S SHOP™

31 Day RISK-FREE TRIAL on any product in this ad.

C Programmers: 7 Ways to Increase Productivity

SORT/MERGE With RECORD SELECTION & OUTPUT REFORMATTING with OPT-TECH SORT

New 3.0 version is even faster and more powerful. Improve your system's performance with OPT-TECH SORT. OPT-TECH includes:

- CALLable and Standalone use
- All major languages
- Variable and fixed length
- Up to 10 sort/select fields
- Autoselect of RAM or disk
- Options: dBASE, BTrieve files
- 1 to 10 files input
- No software max for # records
- Full memory utilization
- All common field types
- Bypass headers, limit sort
- Inplace sort option
- Output = Record or keys

Try what you're using on an XT: 1,000 128 byte records, 10 byte key in 33 seconds.

MSDOS $135

C DYNAMO! WINDOWING: Full C Source, No Royalties POWER WINDOWS and C FUNCTION LIBRARY

Power Windows covers all the bases: overlays, borders, 1-2-3 style or pop-up menus/help windows, zap instantly on/off screen, status lines, horizontal/vertical scrolling, color control or highlighting, word-wrap, files to windows, keyboard to windows. Powerful, easy to use, integrated error messages, thorough documentation. Supports IBM monochrome or color.

C Function Library - includes 325 fundamental functions with readable source and thorough documentation.

No matter what you have, you need these. Best value available. Highly recommended!

Power Windows MSDOS Only $119

C Function Library MSDOS Only $119

db_VISTA First Database Exclusively for C is also Royalty-Free

— *"If you are looking for a sophisticated C programmer's database, this is it."* — - Dave Schmitt, President, Lattice, Inc.

Designed exclusively for C, db_VISTA is a royalty-free programmer's DBMS. Take full advantage of C through ease of use, portability, and efficiency. You optimize for speed and efficient disk storage.

Multiple key records, fast B-tree indexing, virtual memory disk accessing. Tailor db_VISTA to your needs by using only those features you require. Optional dBASE, R:BASE, and ASCII file transfer utilities make moving to db_VISTA a snap.

MSDOS, Unix, Xenix, MacIntosh. Single user Source $459. Object $179. Multiuser Source $929. Object $450

Add Full Modem Control to Your Program Greenleaf Comm Library

Writing and debugging communication programs can be difficult and frustrating. Use stable, reliable and comprehensive existing code. Communicate with remote systems or databases with an asynchronous communications library for C.

Individual transmission and reception ring buffers combine with an interrupt-driven system.

Included are 3 library/object files, 220 functions, 230 page manual, **complete source code.** Library supports Microsoft 3.0, Lattice 3.0, Aztec, and others. Hayes-compatible modem commands, and a complete sample file transfer program.

PCDOS $149

Cross Compiler with COMPLETE SOURCE: QCX

Expand your audience with a cross-compiler and get full source code to the compiler and library. QCX can help you learn techniques for compiler design and parsing while providing you with a low-cost development tool. Hosted on MSDOS, Unix (or Xenix), or Vax VMS and generates ROMable Z80 assembly code. Cross assemblers and linkers are separately available.

QCX supports long integers and single-precision floating point, with only minor restrictions to K&R standard. Arguments to functions are reversed, which allows any function to have a variable number of arguments.

AO, an optional host optimizer, is available for only $125. Buy QCX for $465 and we'll include AO FREE!

Consider the native MSDOS compiler for $125.

MSDOS $465

Even for Small Files: Convenient, Fast Access CBTREE — Only $99

Why spend time writing file management code when you can use consistent, flexible, documented, professional functions? Even multiuser record locking and variable-length records are supported.

Add, delete, and update without needing to reindex. Store keys and record locations in B+trees.

You can access any record or group of records by the value of a user specific key. Search your files from any point, forward or backward.

Full, balanced B-tree support includes use of multiple keys, unlimited number and length of keys.

Use this powerful ISAM, even if you've previously done without.

Learn how to write systems for managing large files by using CBTREE source as a guide. Modify it and transfer it to another operating environment without royalties.

MSDOS $99

Comprehensive C Development Library C-Worthy by Custom Design Systems

C-WORTHY eliminates the writing of routine code and frees you to work on what makes your programs unique. Includes 425 pages of documentation with an in-depth tutorial and source code to a sample program.

A complete, consistent, and interrelated set of subsystems and functions facilitates keyboard handling, background procedures, list manipulation, screen handling, menu management, windowing, error reporting, context-sensitive help, DOS interfacing, and MORE.

Now you can support incompatible machines (like IBM PC, VICTOR 9000, Texas Instruments Professional, NEC APCIII, Wang PC, and HP 150) with the same .EXE file. No recompilation or relinking is required. All machine-dependent aspects of each microcomputer are isolated in a separate runtime overlay file which is loaded into the computer's memory along with the application at runtime. C-Worthy applications can also be executed on networks running Novell's NetWare.

The C-Worthy development utilities Help Librarian, Message Librarian, and Error Librarian allow applications to support alternate languages (like French, German, etc.) with the same source code. C-Worthy's philosophy of program development is revolutionary, encouraging modular design through the use of action procedures as arguments to functions. C-Worthy's unique design approach provides application developers with a consistent and intuitive user interface with features for both novice and advanced users. No royalties. For Lattice C and others.

MSDOS $295

Dear C Programmer:

You want the best development software for your needs. These products will help you:

- Speed your development efforts
- Write even better programs
- Increase productivity
- Reduce your programming frustration

We carry over 100 C compilers, interpreters, support libraries, debuggers, and addons, specifically designed for C programmers using MSDOS or PCDOS. Call one of our knowledgeable consultants - toll free - for details, comparisons, or for one of our specially prepared packets on C.

There is no obligation. You must be completely satisfied with the products you purchase or you will receive a full refund or replacement. You risk nothing with our 31 day risk-free trial on any product in this ad.

Yours for more productive programming,

Bruce W. Lynch

Bruce W. Lynch, President

Call for a catalog, literature, advice and service you can trust

NEW HOURS
8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™
128-D Rockland Street, Hanover, MA 02339
Mass: 800-442-8070 or 617-826-7531

3/86

"You've got everything I've heard of, and much I haven't! . . . Normally, I expect my money to be "fan letter" enough, but you people are SUPER!"
— Shel Hall
Artell Corp.

NGS FORTH

A FAST FORTH, OPTIMIZED FOR THE IBM PERSONAL COMPUTER AND MS-DOS COMPATIBLES.

STANDARD FEATURES INCLUDE:

- 79 STANDARD
- DIRECT I/O ACCESS
- FULL ACCESS TO MS-DOS FILES AND FUNCTIONS
- ENVIRONMENT SAVE & LOAD
- MULTI-SEGMENTED FOR LARGE APPLICATIONS
- EXTENDED ADDRESSING
- MEMORY ALLOCATION CONFIGURABLE ON-LINE
- AUTO LOAD SCREEN BOOT
- LINE & SCREEN EDITORS
- DECOMPILER AND DEBUGGING AIDS
- 8088 ASSEMBLER
- GRAPHICS & SOUND
- NGS ENHANCEMENTS
- DETAILED MANUAL
- INEXPENSIVE UPGRADES
- NGS USER NEWSLETTER

A COMPLETE FORTH DEVELOPMENT SYSTEM.

PRICES START AT $70

NEW◆HP-150 & HP-110 VERSIONS AVAILABLE

NEXT GENERATION SYSTEMS
P.O.BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

Circle **no. 185** on reader service card.

C CHEST

Listing One (Text begins on page 18.)

```
                    Listing 1 -- tree.c
----------------------------------------------------------------------------
#include <stdio.h>

/*      TREE.C: Various binary tree routines:
 *
 *              (C) 1986, Allen I. Holub.  All rights reserved.
 */

typedef struct _n
{
        struct _n    *left;
        struct _n    *right;
        char         *key;
}
LEAF;

char            *Map;
extern  char    *makebitmap();

/*----------------------------------------------------------------------
 * These defines are used by the lr_trav() routine:
 */

#define mark(p)         ( *( (p)->key )  |=  0x80 )
#define marked(p)       ( *( (p)->key )  &   0x80 )
#define unmark(p)       ( *( (p)->key )  &= ~0x80 )
#define visit(root)     printf("%s ", root->key );

/*----------------------------------------------------------------------*/

tree( key, rootp )
char    *key;
LEAF    **rootp;        /* POINTER to (not the contents of) the root    */
{
        /* Non-recursive binary tree search and insert function. If key
         * is in the tree a message to that effect is printed, otherwise
         * a node containing key is inserted into the tree at the
         * proper place.
         */

        LEAF    *root          = *rootp ;
        LEAF    **insert_here =  rootp ;
        LEAF    *malloc();
        int     rel;

        while( root )
        {
                if( (rel = strcmp(key, root->key)) == 0 )
                {
                        printf("key <%s> in tree\n", key );
                        return;
                }
                else
                {
                        insert_here = (rel < 0) ? &root->left : &root->right ;
                        root = *insert_here ;
                }
        }

        if( *insert_here = root = malloc(sizeof(LEAF)) )
        {
                root->right = root->left = NULL;
                root->key   = key;
        }
        else
                printf("Out of memory.\n");
}

/*----------------------------------------------------------------------*/

sinorder( root )
LEAF    *root;
{
        /* A simple recursive in-order traversal, each node is printed
         * with as many tabs to it's left as it is deep in the tree.
         * (if a node is at depth 4 then 4 tabs are printed).
         */

        static int      depth = -1;
        register int    i;

        if( root )
        {
                depth++;

                inorder( root->left );
                for(i = depth; --i >= 0 ; putchar('\t') )
                        ;

                printf( "%s\n", root->key );
                inorder( root->right );
```

```
                depth--;
        }
}

/*----------------------------------------------------------------------
 * inorder():
 *
 * Does an recursive in-order traversal of a binary tree, printing
 * it in graphic form (showing the various pointers). Note that
 * the traverse order is reversed (go right, print root, go left)
 * so that a mirror image of the tree won't be printed (normal
 * traversal would result in the leftmost node of the left subtree
 * being printed first).
 *
 *     | is associated with this depth in the bitmap:
 *          1       2       3
 *          |       |       |
 *          V       V       |
 *                          V
 *          +---1---+
 *          |       |       +---2
 *          |       +---2---+
 *     0---+                        Node number == depth in tree.
 *          |       +---2
 *          +---1---+
 *                  +---2
 */

inorder( root, amleft )
LEAF    *root;          /* Root of current subtree                      */
int     amleft;         /* Root is a left decendant of the parent       */
{
        static int      depth = -1;     /* Current depth in the tree    */
        int             i;

        if( root )
        {
                ++depth;

                if( root->right )
                        inorder( root->right, 0 );
                else
                        setbit( depth+1, Map, 1 );

                for(i = 1; i <= depth ; i++ )
                {
                        printf( i == depth      ? "    +---" :
                                testbit(i,Map)  ? "    |   " :
                                                  "        " );
                }

                printf( "%s%s\n", root->key,
                                root->left || root->right ? "---+" : "" );

                setbit( depth, Map, amleft ? 0 : 1 );

                if( root->left )
                        inorder( root->left, 1 );
                else
                        setbit( depth+1, Map, 0 );

                --depth;
        }
}


/*----------------------------------------------------------------------*/


pline( depth )
{
        int     i;
        for(i = 0; i < depth-1 ; i++ )
                printf( testbit(i,Map) ? "|       " : "        " );
}

/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/

preorder( root, amright )
LEAF    *root;
{
        /* Does a recursive pre-order traversal of a binary tree printing
         * the tree in graphic form. Though this routine is interesting,
         * it is more useful when adapted to multi-way tree traversal
         * for use in such aplications a printing directory trees.
         */

        static int      depth = -1;

        if( root )
        {
                pline( ++depth );
                printf( "%s%s\n", depth ? "+-------" : "" , root->key );

                if( root->right )
```

(continued on next page)

Bill Gates
(Microsoft BASIC)

"If you ever talk to a great programmer, you'll find he knows his tools like an artist knows his paintbrushes."

Programmers at Work
Interviews with 19 of Today's Top Programmers
$14.95 (soft) **$19.95** (hard)
MICROSOFT PRESS
Available wherever fine books are sold.

Circle **no. 212** on reader service card.

Unix-like Tools
ATARI ST and IBM PC

Unix-like Shell
MICRO C-Shell **$49.95**
Aliases, full history (!$, !*, etc.), backquote command substitution, C shell scripts (if-then-else, foreach, while, break, continue, goto). Many Unix-like utilities: cat, chmod, cmp, cp, date, diff, grep, ls, lpr, mkdir, more, mv, pr, printenv, rm, rmdir, sed, setenv, tail, tee, wc.

More Unix-like Tools
MICRO C-Tools **$24.95**

Unix-like Make Utility
MICRO Make **$34.95**
Unix make syntax and options. Works with any compiler, linker. Runs commands directly or uses MICRO C-Shell above.

Atari ST
Real-Time Executive
MICRO RTX **$69.95**
Real-time multitasking operating system kernel for the Atari ST.

VISA
Beckemeyer Development Tools
592 JEAN STREET, NO. 304
OAKLAND, CA 94610
MasterCard

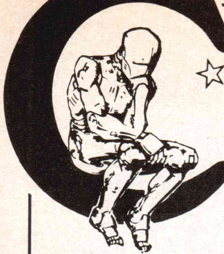
Circle **no. 290** on reader service card.

Gary Kildall
(CPM)

"Somehow we have to break loose from the ways we think about microcomputers if we want to stimulate advances in computers."

Programmers at Work
Interviews with 19 of Today's Top Programmers
$14.95 (soft) **$19.95** (hard)
MICROSOFT PRESS
Available wherever fine books are sold.

Circle **no. 125** on reader service card.



Thinking about C?

Stop Thinking-
Start Programming Today!

SPECIAL INTRODUCTORY OFFER!
C' Prime, **Personal Computing and C,**
Plus Apprentice C.
A $169 value only **$99**

NEW FROM MANX AZTEC!

C' Prime ~~$99~~ **$79**

Never has C been easier to learn. **Manx Aztec** is now offering a complete C system called **C' Prime** at an exceptionally low price. This powerful system includes a Compiler, Linker, Assembler, Editor, Libraries and Object Librarian. **C PRIME** supports a host of third-party software.

C Apprentice ~~$49.95~~ **$39.95**

Learn C quickly with this complete, easy-to-use C language interpreter. Apprentice C includes a complete one-step compiler that executes with lightning speed, an editor, and a run-time system.

NEW FROM ASHTON-TATE!

Personal Computing and C

A detailed, easy-to-understand guide to C programming prepared especially by Ashton-Tate for use with the new Aztec **C' Prime.** Includes chapters on C programming basics, function libraries, data handling, and advanced features, plus a complete money management demonstration program.

To order or for information call:

TECWARE
1-800-TEC-WARE
(In NJ call 201-530-6307)

UNIX is a registered TM of Bell Laboratories. dBase TM Aston-Tate, Inc. MANX AZTEC, C PRIME, Apprentice C TM Manx Software Systems, Inc

Circle **no. 222** on reader service card.

C CHEST

Listing One (Listing continued, text begins on page 18.)

```
                        setbit( depth, Map, 1 );

                preorder( root->left,  0 );
                setbit  ( depth, Map,  0 );
                preorder( root->right, 1 );

                if( amright && !(root->right || root->left) )
                {
                        pline( depth );
                        printf("\n");
                }

                depth--;
        }
}

/*----------------------------------------------------------------------
 * lr_trav() (below) does a non-recursive link-reversal traversal of a
 * binary tree. The algorithm used is:
 *
 * do forever
 *      if( marked( pres ) )
 *              clear mark
 *      else
 *              while( there's a left child )
 *                      preorder visit
 *                      go left
 *
 *              inorder & preorder visit
 *
 *      postorder visit
 *
 *      if( no previous node )
 *              break
 *
 *      if( marked( prev ) )
 *              go up from a right child
 *      else
 *              go up from a left child
 *              inorder visit
 *              set mark
 *              go right
 *
 *   "Go" means to decend one node in the tree, reversing the pointer
 *   to that node so that it points back up where we came from.
 *   If we "go left" then we reverse the the left pointer;
 *   if we "go right" then right pointer is reversed. "Go up" means
 *   return to the previous node and make the pointer point back to
 *   its original location. A node is marked after we have
 *   traversed its left sub-tree. The mark is cleared after we've
 *   traversed both the left and right sub-trees. The high bit of
 *   the "key" field is used to mark the node, you could also add
 *   another field to the structure if you have a numeric key. Only
 *   one bit is needed for the mark.
 */

lr_trav( pres )
LEAF  *pres;
{
        LEAF    *prev = NULL, *next;

        while( 1 )
        {
                if( marked(pres) )
                        unmark( pres );
                else
                {
                        while( next = pres->left )
                        {
                                /* preorder visit */
                                /* goes here      */

                                pres->left = prev;      /* go left        */
                                prev       = pres;
                                pres       = next;
                        }
                        visit( pres );                  /* inorder & pre- */
                                                        /* order visit    */
                }

                /* postorder visit goes here */

                if( !prev )
                        break;

                if( marked(prev) )
                {
                        next        = prev->right;      /* go up from a */
                        prev->right = pres;             /* right child  */
                        pres        = prev;
                        prev        = next;
```

```
                }
                else
                {
                        next      = prev->left;         /* go up from a */
                        prev->left = pres;              /* left child   */
                        pres      = prev;
                        prev      = next;

                        visit( pres );                  /* inorder visit */

                        mark( pres );                   /* mark the node */
                        if( next = pres->right )        /* go right      */
                        {
                                pres->right = prev;
                                prev        = pres;
                                pres        = next;
                        }
                }
        }

        printf("\n");
}

/*-------------------------------------------------------------------*/

main(argc, argv)
char    **argv;
{
        static LEAF     *root = NULL ;
        char            buf[128];

        Map = makebitmap( 128 );

#ifdef MODEL
        for( printf("? "); gets(buf); printf("? ") )
        {
                tree( strsave(buf), &root );
                lr_trav ( root );
                inorder ( root , 0 );
                preorder( root , 0 );
        }
#endif

        while( --argc > 0 )
                tree( *++argv, &root );

        inorder ( root , 0 );
}
```

End Listing One

Listing Two

```
                        Listing 2 -- bitmap.c
-----------------------------------------------------------------------
/*      BITMAP.C        makebitmap, setbit, testbit: bit map manipulation
 *                      routines.
 *
 *      Copyright (c) 1985,1986     Allen I. Holub, all rights reserved.
 *
 *      These routines originally appeared in the June, 1985, C Chest (DDJ
 *      #104). They are reproduced here in a stripped-down version.
 *      Please see that column for more information about how they work.
 */

extern char     *calloc         ( unsigned, unsigned );

typedef char BITMAP;

/*-------------------------------------------------------------------*/

BITMAP    *makebitmap( size )
unsigned  size;
{
        /*  Make a bit map for "size" bits.                          */
        /*  Return a pointer to the map or NULL if we couldn't make it. */

        unsigned *map, numbytes;

        numbytes = (size >> 3) + ((size & 0x07) ? 1 : 0 ) ;

        if( map = (unsigned *) calloc( numbytes + sizeof(unsigned) ,1 ) )
                *map = size;

        return (BITMAP *) map;
}

/* ----------------------------------------------------------------- */

setbit( c, map, val )
unsigned  c, val;
char      *map;
{
        /*   Set bit c in the map to val.                            */
        /*   If c > map size, 0 is returned, else 1 is returned.     */
```

(continued on next page)

Butler Lampson
(Alto PC)

"To hell with computer literacy. It's absolutely ridiculous. Study mathematics. Learn to think. Read. Write."

Programmers at Work

Interviews with 19 of Today's Top Programmers

$14.95 (soft) **$19.95** (hard)

MICROSOFT PRESS

Available wherever fine books are sold.

Circle **no. 126** on reader service card.

ICs PROMPT DELIVERY!!!

SAME DAY SHIPPING (USUALLY)

QUANTITY ONE PRICES SHOWN

OUTSIDE OKLAHOMA: NO SALES TAX

640 Kbyte MOTHERBOARD KITS: Zenith 150: $78.10
IBM PC/XT, Compaq Portable & Plus; hp Vectra

DYNAMIC RAM			
256K	64Kx4	150 ns	$4.37
256K	256Kx1	100 ns	4.87
256K	256Kx1	120 ns	3.35
256K	256Kx1	150 ns	2.95
128K	128Kx1	150 ns	4.55
64K	64Kx1	150 ns	1.40
EPROM			
27512	64Kx8	250 ns	$28.00
27C256	32Kx8	250 ns	7.85
27256	32Kx8	250 ns	5.45
27128	16Kx8	250 ns	3.90
27C64	8Kx8	200 ns	5.10
2764	8Kx8	250 ns	3.65
STATIC RAM			
43256L-12	32Kx8	120 ns	$50.00
6264LP-15	8Kx8	150 ns	3.10
6116LP-3	2Kx8	150 ns	2.10

8087-2 $175.00
8087-3 $125.00
V20/V30 8 Mhz $ 17.50
80287-8 8 Mhz $365.00

OPEN 6½ DAYS: WE CAN SHIP VIA FED-EX ON SAT.

NO EXTRA COST FOR FED-EX SAT DELIVERY ON ORDERS RECEIVED BY
Th: Std Air $6/4 lbs
Fr: P-One $13/2 lbs

MasterCard/VISA or UPS CASH COD

Factory New, Prime Parts µP∞

MICROPROCESSOR UNLIMITED, INC.
24,000 S. Peoria Ave.,
BEGGS, OK. 74421
(918) 267-4961

Prices shown above are for May 26, 1986

Please call for current prices. Prices subject to change. Please expect higher or lower prices on some parts due to supply & demand and our changing costs. Shipping & insurance extra. Cash discount prices shown. Orders received by 6 PM CST can usually be delivered to you **by the next morning, via Federal Express Standard Air @ $6.00, or Priority One @ $13.00!**

Circle **no. 105** on reader service card.

Jonathan Sachs
(Lotus 1-2-3)

"The rate of innovation is rather slow. There are only a few really new ideas every decade."

Programmers at Work

Interviews with 19 of Today's Top Programmers

$14.95 (soft) **$19.95** (hard)

MICROSOFT PRESS

Available wherever fine books are sold.

Circle **no. 215** on reader service card.

Listing Two (Listing continued, text begins on page 18.)

```

        if(  c >= *(unsigned *)map )               /* if c >= map size */
                return 0;

        map += sizeof(unsigned);                   /* Skip past size   */

        if( val ) map[c >> 3]  |=    1 << (c & 0x07)  ;
        else      map[c >> 3]  &=  ~(1 << (c & 0x07)) ;

        return( 1 );
}

/* ------------------------------------------------------------------ */

testbit( c, map )
unsigned c;
char     *map;
{
        /*   Return 1 if the bit corresponding to c in map is set.      */
        /*   0 if it is not.                                             */

        if( c >= *(unsigned *)map )
                return 0;

        map += sizeof(unsigned);

        return(  map[ c >> 3 ]  &  (1 << (c & 0x07)) );
}
```

End Listings

IQ©LISP

THE CLOSEST THING TO *COMMON LISP* AVAILABLE FOR YOUR PC

RICH SET OF DATA TYPES
Bignums, for high precision arithmetic
8087 support, for fast floating point
Arrays, for multidimensional data
Streams, for device-independent i/o
Packages, for partitioning large systems
Characters, strings, bit-arrays

FULL SET OF CONTROL PRIMITIVES
flet, labels, macrolet, for local functions
if, when, unless, case, cond, for conditionals
Keyword parameters, for flexibility
Multiple-valued functions, for clarity
Flavors, for object-oriented programming
Stacks, for coroutining
Closures, for encapsulation

LARGE COMPLEMENT OF FUNCTIONS
Mappers, for functional programming
format, for output control
sort, for user-specified predicates
Transcendental floating point functions
String handling functions
Over *400* functions altogether

APPLICATION SUPPORT
Save and restore full environments
User-specified initializations
Assembly language interface

HARDWARE REQUIREMENTS
8088 or 8086 CPU, MSDOS Operating System
390K RAM or more

IQCLISP
5¼'' diskettes and manual $300.00
Foreign orders add $30.00 for airmail.
U.S. Shipping included for prepaid orders.

Integral Quality
P.O. Box 31970
Seattle, Washington 98103
(206) 527-2918
Washington State residents add sales tax.
VISA and MASTERCARD accepted.

EXTENDABILITY
defstruct, to add data types
Macros, to add control constructs
Read macros, to extend the input syntax
Extendable arithmetic system
Customizable window system

DEBUGGING SUPPORT
step, for single-stepping
trace, for monitoring
break, for probing
inspect, for exploring
Flexible error recovery
Customizable pretty-printer

MSDOS INTERFACE
Random access files
Hierarchical directory access
MSDOS calls

DOCUMENTATION
On-line documentation of functions
apropos
300-page indexed reference manual

Product Information

Free!

Dr. Dobb's Journal of Software Tools

July 1986 #117 **Expiration Date:** October 31, 1986

Name ______________________ Title ______________________

Company ______________________ Phone ______________________

Address ______________________

City/State/Zip ______________________

Please circle one letter in each category:

I. My work is performed:
- A. for in-house use only.
- B. for other companies.
- C. for end users/retailers.
- D. in none of the above areas.

II. My primary job function:
- A. Software Project Mgmt/Spvr
- B. Hardware Project Mgmt/Spvr
- C. Computer Consultant
- D. Corporate Management
- E. Other

III. My company department performs:
- A. software development.
- B. computer system integration.
- C. computer manufacturing.
- D. computer consulting.
- E. computer research.
- F. none of the above.

IV. This inquiry is for:
- A. a purchase within 1 month.
- B. a purchase within 1 to 6 months.
- C. product information only.

V. Corporate Purchase Authority:
- A. Final Decision-maker
- B. Approve/Recommend
- C. No Influence

VI. Personal Computer Users at my Jobsite:
- A. 10,000 or more
- B. 500 to 9,999
- C. 100 to 499
- D. 10 to 99
- E. less than 10

VII. On average, I advise others about computers:
- A. more than once per day.
- B. once per day.
- C. once per week.
- D. less than once per week.

VIII. In my job function, I:
- A. design software and/or write code.
- B. design software.
- C. write code.
- D. don't design software or write code.

A Reader Service number appears on each advertisement. Circle the corresponding numbers below for more info.

001	002	003	004	005	006	007	008	009
010	011	012	013	014	015	016	017	018
019	020	021	022	023	024	025	026	027
028	029	030	031	032	033	034	035	036
037	038	039	040	041	042	043	044	045
046	047	048	049	050	051	052	053	054
055	056	057	058	059	060	061	062	06
064	065	066	067	068	069	070	071	07
073	074	075	076	077	078	079	080	08
082	083	084	085	086	087	088	089	09
091	092	093	094	095	096	097	098	09
100	101	102	103	104	105	106	107	10
109	110	111	112	113	114	115	116	11
118	119	120	121	122	123	124	125	12
127	128	129	130	131	132	133	134	13
136	137	138	139	140	141	142	143	14
145	146	147	148	149	150	151	152	15
154	155	156	157	158	159	160	161	16
163	164	165	166	167	168	169	170	17
172	173	174	175	176	177	178	179	18
181	182	183	184	185	186	187	188	18
190	191	192	193	194	195	196	197	19
199	200	201	202	203	204	205	206	20
208	209	210	211	212	213	214	215	21
217	218	219	220	221	222	223	224	22
226	227	228	229	230	231	232	233	23
235	236	237	238	239	240	241	242	24
244	245	246	247	248	249	250	251	25
253	254	255	256	257	258	259	260	26
262	263	264	265	266	267	268	269	27
271	272	273	274	275	276	277	278	27
280	281	282	283	284	285	286	287	28
289	290	291	292	293	294	295	296	29
298	299	**999**						

Circle 999 to start a 12 month subscription at the price of $25.00

Postage Paid!

Thank You!
Dr. Dobb's greatly appreciates your responses to questions I through VIII.

Product Information

Free!

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
P.O. Box 2157
Clinton, Iowa 52735-2157

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
P.O. Box 2157
Clinton, Iowa 52735-2157

Thank You!
Dr. Dobb's greatly appreciates your responses to questions I through VIII.

Reader Service number appears on each
vertisement. Circle the corresponding
mbers below for more info.

002	003	004	005	006	007	008	009	
011	012	013	014	015	016	017	018	
020	021	022	023	024	025	026	027	
029	030	031	032	033	034	035	036	
038	039	040	041	042	043	044	045	
047	048	049	050	051	052	053	054	
056	057	058	059	060	061	062	063	
065	066	067	068	069	070	071	072	
074	075	076	077	078	079	080	081	
083	084	085	086	087	088	089	090	
092	093	094	095	096	097	098	099	
101	102	103	104	105	106	107	108	
110	111	112	113	114	115	116	117	
119	120	121	122	123	124	125	126	
128	129	130	131	132	133	134	135	
137	138	139	140	141	142	143	144	
146	147	148	149	150	151	152	153	
155	156	157	158	159	160	161	162	
164	165	166	167	168	169	170	171	
173	174	175	176	177	178	179	180	
182	183	184	185	186	187	188	189	
191	192	193	194	195	196	197	198	
200	201	202	203	204	205	206	207	
209	210	211	212	213	214	215	216	
218	219	220	221	222	223	224	225	
227	228	229	230	231	232	233	234	
236	237	238	239	240	241	242	243	
245	246	247	248	249	250	251	252	
254	255	256	257	258	259	260	261	
263	264	265	266	267	268	269	270	
272	273	274	275	276	277	278	279	
281	282	283	284	285	286	287	288	
290	291	292	293	294	295	296	297	
299	**999**							

le 999 to start a 12 month subscription at price of $25.00

Dr. Dobb's Journal of Software Tools

July 1986 #117 **Expiration Date:** October 31, 1986

Name ______________ Title ______________

Company ______________ Phone ______________

Address ______________

City/State/Zip ______________

Please circle one letter in each category:

I. My work is performed:
A. for in-house use only.
B. for other companies.
C. for end users/retailers.
D. in none of the above areas.

II. My primary job function:
A. Software Project Mgmt/Spvr
B. Hardware Project Mgmt/Spvr
C. Computer Consultant
D. Corporate Management
E. Other

III. My company department performs:
A. software development.
B. computer system integration.
C. computer manufacturing.
D. computer consulting.
E. computer research.
F. none of the above.

IV. This inquiry is for:
A. a purchase within 1 month.
B. a purchase within 1 to 6 months.
C. product information only.

V. Corporate Purchase Authority:
A. Final Decision-maker
B. Approve/Recommend
C. No Influence

VI. Personal Computer Users at my Jobsite:
A. 10,000 or more
B. 500 to 9,999
C. 100 to 499
D. 10 to 99
E. less than 10

VII. On average, I advise others about computers:
A. more than once per day.
B. once per day.
C. once per week.
D. less than once per week.

VIII. In my job function, I:
A. design software and/or write code.
B. design software.
C. write code.
D. don't design software or write code.

Product Information

Free!

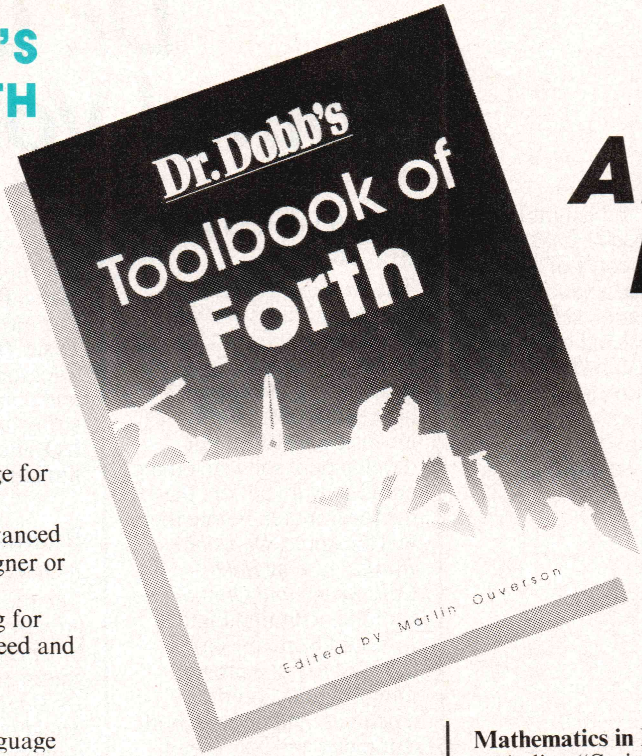
Postage Paid!

Product Information

Free!

Dr. Dobb's Catalog

NEW! DDJ'S JULY FORTH ISSUE SPECIAL!



Announcing Dr. Dobb's Toolbook of Forth

Is Forth the language for you?

It is if you're an advanced user, a systems designer or an applications programmer looking for flexibility, power, speed and minimal program development time.

Forth is also the language for you if you're a beginner who wants to learn more about your computer than you can with a conventional "teaching" language. Whatever your level of expertise, Forth may be for you simply because it is a truly interesting language.

Now, *Dr. Dobb's Journal of Software Tools*, presents Dr. Dobb's Toolbook of Forth.

This comprehensive collection of useful Forth programs and tutorials contains expanded and revised versions of *DDJ's* best Forth articles, along with new Forth material. In addition to the practical code and tutorials, you'll glean important insights about the potential of this increasingly popular language from the many in-depth discussion of advanced Forth topics.

You'll find sections on:

Forth—the Language, including "The Forth Philosophy," Teaching Forth as a First Language" and Forth-83 and Vocabularies"

Forth Programs, including "GO in Forth," "Elements of a Forth Data-Base Design," "The Forth Sort," "SEND & RECV," "Interface for a Mouse," "Relocating Loader in Forth," "Forth Decompiler," "Screen-Oriented Editor Re-Visited," "Evolution of a Video Editor," "H-19 Screen Editor" and "The Conference Tree"

Mathematics in FORTH including "Series Expansion in Forth," "FORTH Floating-Point Package," "Signed Integer Division"

Modifications/Extensions, including "A Proposal for Strings in Forth," "Non-Deterministic Control Words," "Some Forth Coding Standards," "Towards a More Writable Forth Syntax"

Implementing FORTH, including "Forth and the Motorola 68000," "A 68000 Forth Assembler," "A Forth Assembler for the 6502," "Z8000 Forth"

You'll also find Appendices that will help you convert fig-Forth to Forth-83, and tell you how to stay up-to-date on the latest developments and refinements of this popular language.

The screens in the book are also available on disk as Ascii files. Receive *Dr. Dobb's Toolbook of Forth*, along with the software on disk, together for only $39.95.

Item #030 Dr. Dobb's Toolbook of Forth $22.95

Item #031 Toolbook of Forth with disk $39.95

Please specify MS/PC DOS, Apple II, Macintosh, or CP/M. For CP/M disks, specify Osborne or 8" SS/SD.

Please allow 6 weeks for delivery of the **Toolbook of Forth**.

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE 1-800-528-6050 EXT. 4001 and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 216

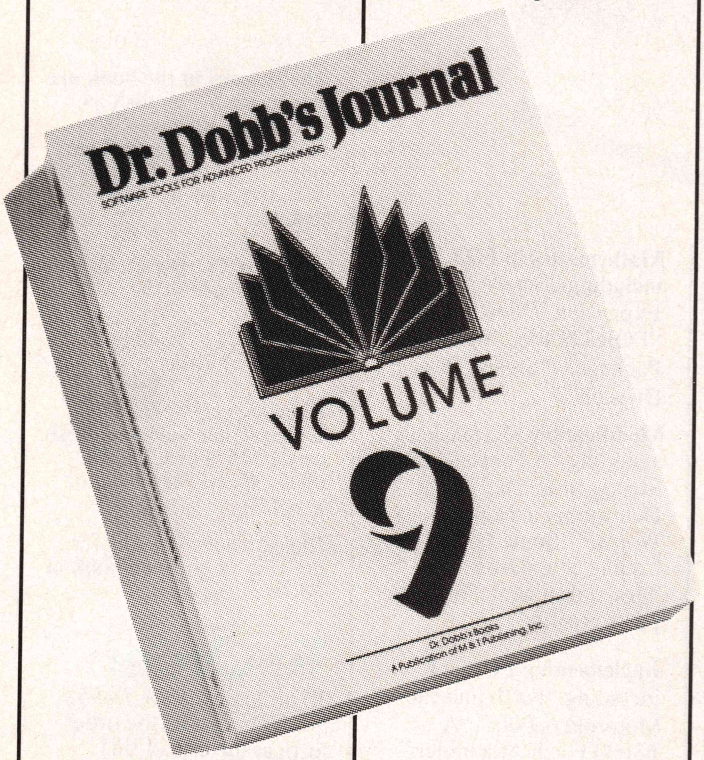
M&T BOOKS

Dr. Dobb's Bound Volumes

Programs by the Pound

Announcing: Dr. Dobb's Bound Volume 9

Over 1000 pages of listings and text. The entire 1984 editorial contents of *Dr. Dobb's Journal of Software Tools.*



Bound Volume 9: 1984

Item #020B

Shaping things to come. In 1984 new Editor-in-chief Mike Swaine brought his interests in advanced technology to *Dr. Dobb's Journal.* We presented the concepts behind Prolog and published an expert system for weather prediction. We learned Modula-2, and taught Forth to talk to a 68000, to MS DOS, and to the people of China. We examined a new language implementation called Turbo Pascal and extended implementations of C with a preprocessor, a library, Tony Skjellum's tricks, and Allen Holub's Grep. We published two powerful encryption systems, telecommunications protocols, floating-point benchmark results, and an issue devoted to the internals of Unix. And Ray Duncan, Bob Blum, and Dave Cortesi were on hand with their fascinating columns.

Bound Volume 1: 1976

Item #013

The working notes of a technological revolution. Programmers from Defense laboratory systems analysts to kitchen-table entrepreneurs worked for the intrinsic rewards to put development software on the brand-new invention, the microcomputer. Before there was an Apple, *Dr. Dobb's Journal of Tiny Basic Calisthenics and Orthodontia* (subtitle: Running Light without Overbyte) was founded to put a programming language on the machines, and became both chronicler and instrument of the revolution. In this first-year volume: Tiny Basic, the first word on CP/M, notes on building an IMSAI, floating-point and timer routines.

Bound Volume 2: 1977

Item #014

Running light without overbyte. By year two, *Dr. Dobb's* formula was concocted: tough questions and serious technical issues handled with enthusiasm, wit, and scant reverence for the accepted answers. Source code. Tools for programmers. Respect for tight programming. *Dr. Dobb's Journal* readers shared insights on warping the Intel 8080 into a computer CPU, and *Dr. Dobb's* published a complete operating system for the chip. A motley crop of computers and software products were popping up, and *Dr. Dobb's* investigated: the Heath H-8, the KIM-1, the Alpha Micro, MITS Basic, Poly Basic, and Lawrence Livermore Labs Basic. *Dr. Dobb's* introduced Pilot for microcomputers and published tips on doing string handling, high-speed I/O, and turtle graphics in limited memory.

Bound Volume 3: 1978

Item #015

The roots of Silicon Valley growth. In 1978 Steve Wozniak and other programmers were publishing in *Dr. Dobb's Journal* code that would help them grow multi-million-dollar computer companies. The proposed S-100 bus standard was hashed out in *Dr. Dobb's* pages. *Dr. Dobb's* contributors began to speak more in terms of technique than of specific implementations as the industry began to diversify. Languages covered in depth included SAM76, Pilot, Pascal, and Lisp.

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001

and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 216

M&T BOOKS

Dr. Dobb's Bound Volumes

Programs by the Pound

Bound Volume 4: 1979
Item #016

In the midst of the Gold Rush. Three years before IBM would release its PC, a thriving, rough-and-tumble personal computer industry existed. Fortunes had been made and lost, the effective power of the machine multiplied a hundredfold. By 1979 some stability had even emerged; one could speak of the processors that had proven longevity as microcomputer CPUs: the 8080, the Z80, the 6800, and the 6502. *Dr. Dobb's Journal* focused on the best ways to use these processors, with algorithms, tips, and code for 8- to 16-bit conversion, pseudo-random number generation, micro-to-mainframe connections, telecommunications, and networking. And lots of useful code.

Bound Volume 5: 1980
Item #017

The preeminence of CP/M and the rise of C. More than any other magazine, *Dr. Dobb's Journal* was responsible for the spread of CP/M and C on microcomputers. Both of those movements began in 1980. *Dr. Dobb's* all-CP/M issue, including Gary Kildall's history of CP/M, sold out within weeks of publication. This was the year of Ron Cain's original Small C compiler, of a CP/M-oriented C interpreter, CP/M-to-UCSD Pascal file conversion techniques, and of a greater concern in *Dr. Dobb's* with software portability.

Bound Volume 6: 1981
Item #018

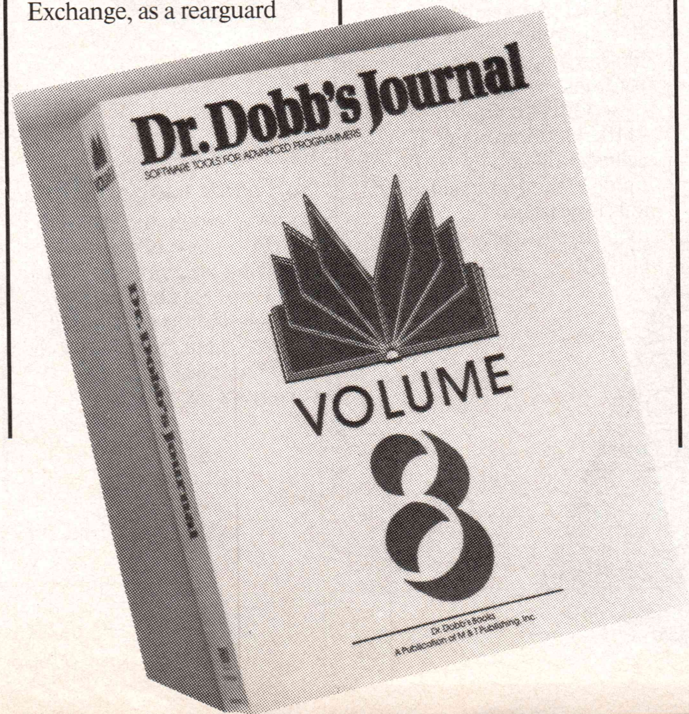
The first of Forth. 1981 saw *Dr. Dobb's* first all-Forth issue (now sold out), along with an emphasis on CP/M, C, telecommunications, and new languages. David Cortesi began "Dr. Dobb's Clinic," one of the magazine's most popular features. Highlights included information on PCNET, the Conference Tree, the Electronic Phone Book, Tiny Basic for the 6809, writing your own compiler, and a systems programming language.

Bound Volume 7: 1982
Item #019

Legitimacy. By 1982 IBM had become a player in the personal computer game and was changing the rules. New microprocessors arrived, the first designed specifically to serve as personal computer CPUs. In *Dr. Dobb's Journal* Dave Cortesi published the first serious comparison of MS DOS and CP/M-86. *Dr. Dobb's* started two new columns: the CP/M Exchange, as a rearguard maneuver to ensure that good tools for CP/M programmers would continue to be developed and circulated, and the 16-Bit Software Toolbox to investigate the 8088/86 and other new microprocessors. We published code for the 68000 and Z8000 processors, and looked ahead, in a provocative essay, to fifth-generation computers.

Bound Volume 8: 1983
Item #020

Power tools. Personal computers were proving themselves to be true professional software development tools by 1983, the year in which Jim Hendrix completed his "canonical" version of Small C in *Dr. Dobb's Journal*. *Dr. Dobb's* published more 68000 and 8088 code, and as the memory limitations relaxed, the magazine's commitment to tight code let it shoehorn impossibly large systems into memory. Small C was just one of the major software products published in their entirety in *Dr. Dobb's* pages that year; there were Ed Ream's RED screen editor and a version of the Ada language called Augusta.



Buy the complete set and save 15%

If you buy all nine volumes, covering the entire editorial content of ***Dr. Dobb's Journal of Software Tools*** from the first issue in 1976 through 1984, you pay just $225. That's a 15% discount and over $40 off the combined individual prices. To order the complete set of Bound Volumes 1 through 9, ask for item #020C.

Item #013	$27.75	Vol. 1
Item #014	$27.75	Vol. 2
Item #015	$27.75	Vol. 3
Item #016	$27.75	Vol. 4
Item #017	$27.75	Vol. 5
Item #018	$27.75	Vol. 6
Item #019	$30.75	Vol. 7
Item #020	$31.75	Vol. 8
Item #020B	$35.75	Vol. 9

All 9 volumes
Item #020C $225.00

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001

and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 216

M&T BOOKS

A collection of powerful tools for C software developers, including books, software on disk, and reference materials from the publisher of Dr. Dobb's Journal of Software Tools

Dr. Dobb's Complete C Toolbox

From M&T Publishing and Brady Communications...

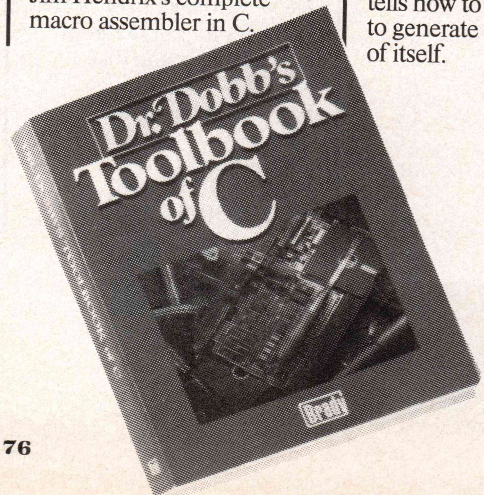
Dr. Dobb's Toolbook of C

Item #005

The **Toolbook** contains over 700 pages of C material, including articles by such C experts as Kernighan and Ritchie, Cain and Hendrix, Skjellum and Holub. The level is sophisticated and pragmatic, appropriate for professional C programmers.

The most valuable part of the **Toolbook** to many will be the hundreds of pages of useful C source code, including a complete compiler, an assembler, and text-processing utilities. The accompanying text explains, in the programmers' own words, why they did what they did.

Dr. Dobb's Journal of Software Tools introduced a generation of personal computer programmers to the C programming language, and all the best C articles and code published in *Dr. Dobb's* over the years is included and updated in the **Toolbook,** including Ron Cain's original Small C article and articles from sold-out issues. But the **Toolbook** also includes material never before published, including Jim Hendrix's complete macro assembler in C.



Dr. Dobb's offers the **Toolbook** in a **special hard-bound edition** for just $29.95. You'll find:

Jim Hendrix's famous **Small C Compiler** and **New Library for Small C** (both also available on disk),

NEW: Hendrix's **Small Mac: An Assembler for Small C** and **Small Tools: Programs for Text Processing** (both also available on disk),

All of Tony Skjellum's **C Programmer's Notebook** columns distilled by Tony into one thought-provoking chapter.

Also from M&T Publishing and Brady Communications...

The Small C Handbook

Item #006 or #006A

Jim Hendrix's **Small-C Handbook** is the reference book on his Small-C compiler. In addition to describing the operation of the compiler, the book contains complete source listings to the compiler and its library of arithmetic and logical routines.

A perfect companion to the Hendrix **Small-C compiler** offered by *Dr. Dobb's* on disk, the **Handbook** even tells how to use the compiler to generate a new version of itself.

While both the **Handbook** and the **Toolbook** provide documentation for the Small-C compiler, the **Handbook** contains a more detailed discussion and is available with an addendum for the MS/PC DOS version.

The **Handbook,** Item #006, is just $17.95. Jim Hendrix has ported the compiler to the MS/PC DOS environment since the **Handbook** was printed, and the **Handbook** plus his **MS/PC DOS Handbook Addendum,** Item #006A, is $22.95.

Dr. Dobb's C Software Tools on Disk

To complement the **Toolbook,** *Dr. Dobb's* also offers the following programs on disk. Full C source code and documentation is included. Except where indicated, both CP/M and MS/PC DOS versions are available.

Small-C Compiler

Item #007

Jim Hendrix's Small-C Compiler is the most popular piece of software ever published in *Dr. Dobb's* 10-year history. Like a home-study course in compiler design, the **Small-C Compiler** and **Small-C Handbook** provide everything you need but the computer for learning how compilers are constructed, and for learning C at its most fundamental level.

While both the **Handbook** and the **Toolbook** provide documentation for the Small-C compiler, the **Handbook** contains a more detailed discussion and is available with an addendum for the MS/PC DOS version. The **MS/PC DOS Small-C Handbook Addendum** is recommended in addition to the **Handbook** for MS DOS or PC DOS users.

The **Small-C Compiler** is available for $19.95 in either the CP/M or the MS/PC DOS version.

To Order:

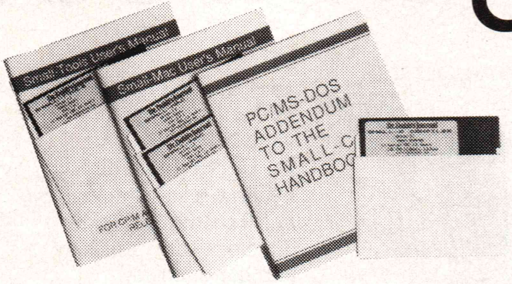
To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE 1-800-528-6050 EXT. 4001 and refer to product item number, title, and disk format.

For customer service questions, CALL M&T PUBLISHING, INC. 415-366-3600 EXT. 216

M&T BOOKS

Dr. Dobb's Complete C Toolbox



Small-Mac: An Assembler for Small-C

Item #012A

Small-Mac is an assembler designed to stress simplicity, portability, adaptability, and educational value. The package features a simplified macro facility, C language expression operators, object file visibility, descriptive error messages, and an externally-defined machine instruction table. You get the macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and a utility to dump relocatable files.

Small-Mac is available with documentation for $29.95. For CP/M systems only.

Small-Tools: Programs for Text Processing

Item #010A

A package of programs performing specific, modular operations on text files, including: editing; formatting; sorting; merging; listing; printing; searching; changing; transliterating; copying; concatenating; encrypting and decrypting; replacing spaces with tabs and tabs with spaces; counting characters, words, or lines; and selecting printer fonts. Supplied in source code form so you can select and adapt these tools to your own purposes.

Small-Tools is available with documentation for $29.95. For CP/M or MS/PC DOS systems.

Special Packages — 20% Off

Now for almost 20% off the combined individual product prices, you can order a complete set of *Dr. Dobb's* C programming tools for your CP/M or MS/PC DOS system.

C Package for CP/M

Item #005A

Ordered individually, these items would cost about $120. If you order the CP/M C package, you'll get ***Dr. Dobb's Toolbook,*** the **Small-C Handbook**, the **Small-C Compiler** on disk, the **Small-Mac** assembler on disk with documentation in the **Small-Mac Manual,** the **Small-Tools** text-processing programs on disk with documentation in the **Small-Tools Manual,** all for just $99.95.

C Package for MS/PC DOS

Item #005B

These items would cost over $100 if purchased individually.

If you order the MS/PC DOS C package, you'll get ***Dr. Dobb's Toolbook,*** the **Small-C Handbook** with the **MS/PC DOS Addendum,** the **Small-C Compiler** on disk, the **Small-Tools** text-processing programs on disk with documentation in the **Small-Tools Manual,** all for just $82.95.

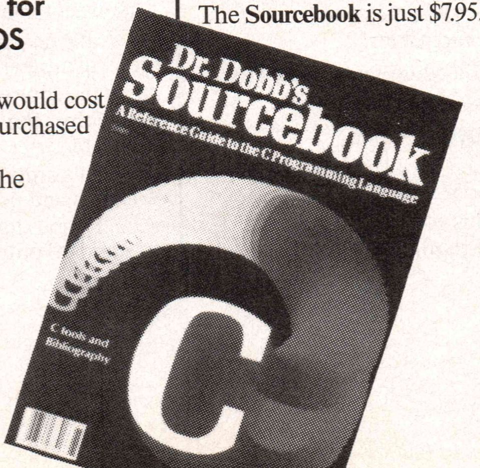
Dr. Dobb's Sourcebook: A Reference Guide to the C Programming Language

Item #004

Products and services for C programmers are appearing at so rapid a rate that it's all but impossible to keep up with them. *Dr. Dobb's* presents this handy guide to the who, what, when, where, and why of C. A comprehensive reference manual to new information, products, and services, the **Sourcebook** contains:

- A bibliography of over 300 articles and books on the C language;
- A descriptive list of products for C programmers: compilers, editors, interpreters, and utilities;
- A list of C-related services: classes, seminars, and on-line services.

The **Sourcebook** is just $7.95.



Dr. Dobb's Sourcebook: A Reference Guide to the C Programming Language Item #004	$ 7.95
Dr. Dobb's Toolbook for C Item #005	$29.95
The Small-C Handbook Item #006	$17.95
The Small-C Handbook and MS/PC-DOS Addendum Item #006A	$22.95
Small-C Compiler disk Item #007	$19.95
Small Tools: Programs for Text Processing disk Item #010A	$29.95
Small Mac: An Assembler for Small-C disk (For CP/M only.) Item #012A	$29.95
CP/M C Package Item #005A	$99.95
MS/PC DOS C Package Item #005B	$82.95

For CP/M disks, please specify one of the following formats: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD.

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE 1-800-528-6050 EXT. 4001 and refer to product item number, title, and disk format.

For customer service questions, CALL M&T PUBLISHING, INC. 415-366-3600 EXT. 216

M&T BOOKS

Dr. Dobb's Catalog

A UNIX-like Shell and Utility Package for MS-DOS

Only $29.95 each! Both for only $50 — save over 15%!

Includes complete source code and documentation.

A UNIX-like Shell for MS-DOS and A UNIX-like Utility Package for MS-DOS

by *Dr. Dobb's* C-Chest columnist, ALLEN HOLUB

The Shell

An MS-DOS implementation of the most often used parts of the UNIX C shell. This package includes an executable version of the shell, along **with complete C source code** and full documentation. Supported features are:

Editing Command line editing with the cursors is supported. The line is visible as you edit it.

Aliases Can be used to change the names of commands or as very fast memory resident batch files.

History The ability to execute a previous command again. The command can be edited before being executed.

Shell Variables Macros that can be used on the command line.

Nested batch files A batch file can call another batch file like a subroutine. Control is passed to the second file and then back to the first one when the second file is finished. DOS doesn't have this capability.

UNIX-like syntax Slash (/) used as a directory separator, minus (—) as a switch designator. A 2048 byte command line is supported. Command line wild card expansion. Multiple commands on a line.

The shell also supports redirection of standard input, standard output, and standard error.

This version corrects several bugs found in the original version printed in *Dr. Dobb's Journal*, December 1985 through March 1986 issues. It runs on IBM-PC's and compatibles.

Add additional features to the Shell with

/util

A UNIX-like Utility Package for MS-DOS

This collection of utility programs for MS-DOS includes updates of the highly acclaimed Dr. Dobb's articles *Grep: A UNIX-Like Generalized Regular Expression Processor, Ls* from *Dr. Dobb's* C Chest column, and *Getargs* from DDJ's C Chest.

Source code is included and all programs (and most of the utility subroutines) are fully documented in a UNIX-style manual. You'll find executable versions of:

cat A file concatenation and viewing program
cp A file copy utility
date Prints the current time and date
du Prints amount of space available and used on a disk
echo Echoes its arguments to standard output
grep Searches for a pattern defined by a regular expression
Ls Gets a sorted directory
mkdir Creates a directory
mv Renames a file or directory. Moves files to another directory.
p Prints a file, one page at a time
pause Prints a message and waits for a response
printenv Prints all the environment variables
rm Deletes one or more files
rmdir Deletes one or more directories
sub Text substitution utility. Replaces all matches of a regular expression with another string.

Order The Shell and /util together for only $50! SAVE OVER 15%!

Item #160 The Shell	$29.95
Item #161 /util	$29.95
Item #162 Shell/util Package	$50.00

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE 1-800-528-6050 EXT. 4001 and refer to product item number, title, and disk format.

For customer service questions, CALL M&T PUBLISHING, INC. 415-366-3600 EXT. 216

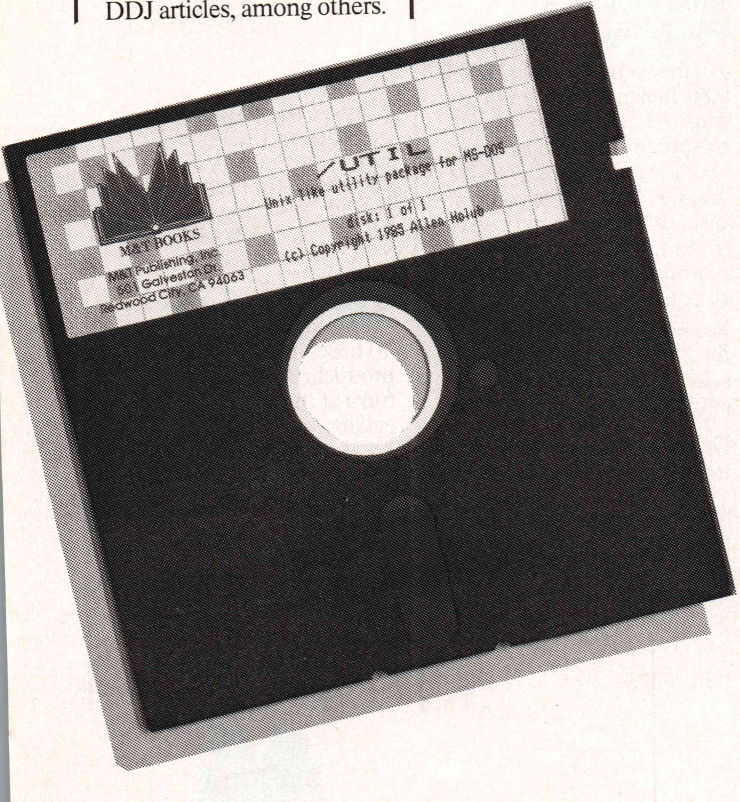
M&T BOOKS

Dr. Dobb's Listings On Disk

Finally! You've Asked For It For Years and Here It Is!

Dr. Dobb's Listings On Disk

Dr. Dobb's Journal of Software Tools has always provided its readers with valuable code. Now, as a useful adjunct to the magazine, DDJ offers the additional value and convenience of selected listings on disk! ***Dr. Dobb's Listings #186*** is a collection of listings from Dr. Dobb's January 1986 issue, through the April 1986 issue. In this first of three ***Dr. Dobb's Listings*** disks for 1986 you'll find listings from the following DDJ articles, among others.



From Issue #111 January 1986

***A Simple OS for Real-time Applications; 68000 assembly language techniques for an operating system kernel* by DDJ editor Nick Turner

***Exec calls and Fortran; a technique allowing execution of a user or system task from a user program* from DDJ's 16-Bit Software Toolbox, by Robert Sypek

***32-bit Square Roots; An 8086 assembly-language routine for 32-bit square roots* by Michael Barr

From Issue #112 February 1986

***Fast Integer Powers for Pascal; An implementation of the fastest-known algorithm for the computation of integer powers* by Dennis E. Hamilton

***Data Abstraction with Modula-2; Construction of a priority queue in Modula-2* by Bill Walker and Stephen Alexander

***Learning Ada on a Micro; A draw poker program in Ada* by Do-While Jones

***Fast IBM PC graphics routines* from DDJ's 16-Bit Software Toolbox, by Dan Rollins

From Issue #113 March 1986

***Recursive Bose-Nelson Sort; An alternative to Joe Celko's September 1985 sort routine* by R. J. Wissbaum

***A Variable-Metric Minimizer; A C program for minimizing arbitrary functions* by Joe Marasco

***Concurrency and Turbo Pascal; An approach to implementing coroutines in Pascal* by Ernest Bergmann

***Speeding MS DOS Disk Access; Programs to test disk-access speed* by Greg Weissman

***Square Roots on the NS32000; Comparable square root routines in C and assembly language for National Semiconductor's 32000 family* by Richard Campbell

From Issue #114 April 1986

***Boca Raton Inference Engine; Lisp, Prolog, and Expert-2 techniques and code* by Robert Brown

Item #170 $14.95
Dr. Dobb's Listings #186
Please specify MS/DOS, Macintosh, or CP/M. For CP/M disks, please specify one of the following formats: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8″ SS/SD.

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or CALL TOLL-FREE 1-800-528-6050 EXT. 4001 and refer to product item number, title, and disk format.

For customer service questions, CALL M&T PUBLISHING, INC. 415-366-3600 EXT. 216

M&T BOOKS

Dr. Dobb's Z80 Toolbook

David E. Cortesi longtime Dr. Dobb's columnist and author of *Inside CP/M* brings you —

Dr. Dobb's Z80 Toolbook

Here's all you need to write your own Z80 assembly language programs for only $25!

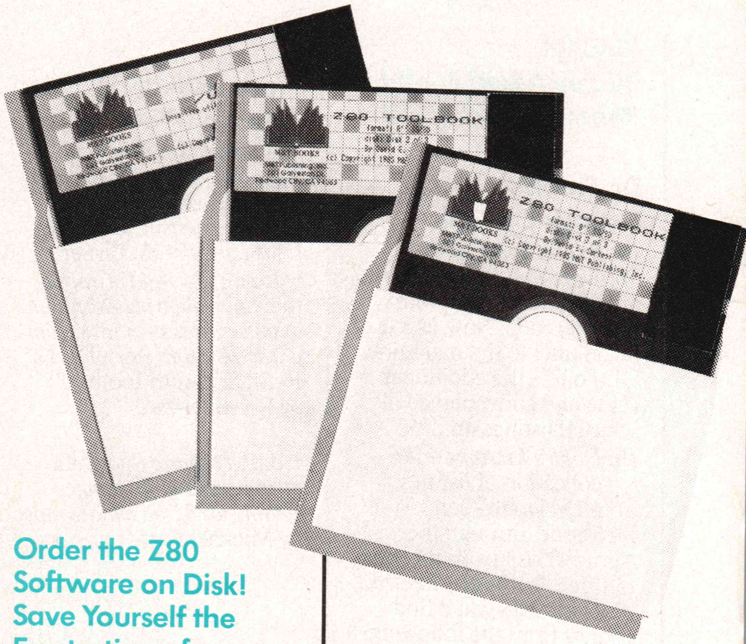
Do you use CP/M? Do you feel as if the only part of the computer industry that has **not** abandoned you is your own Z80 computer? It keeps on working, but when you need programs for it, you have to write them yourself. When you do, you quickly find that while Pascal or BASIC is okay for some things, there is often no substitute for the speed, small size, and flexibility of an assembly language program.

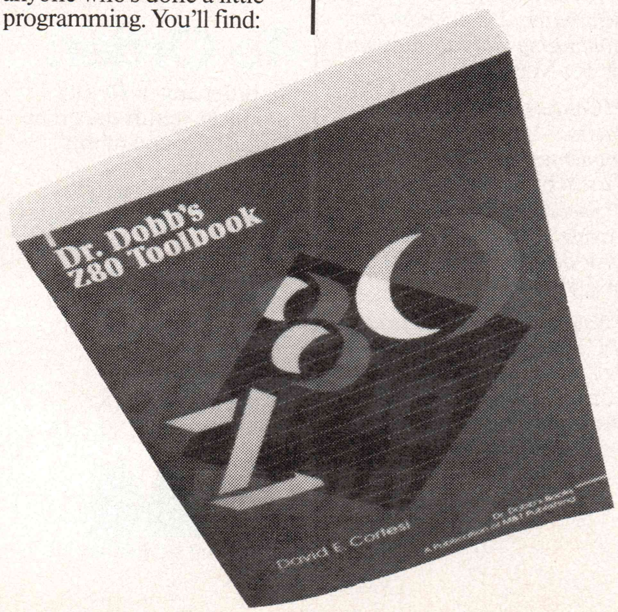
Dr. Dobb's Z80 Toolbook puts the power of assembly language in the hands of anyone who's done a little programming. You'll find:

****A method of designing programs and coding them in assembly language.** Cortesi will take you on a walk through the initial specifications, designing an algorithm and writing the code. He demonstrates this method in the construction of several complete, useful programs.

****A complete, integrated toolkit of subroutines** for arithmetic, for string-handling, an for total control of the CP/M file system. They bring the ease and power of a compiler's runtime library to your assembly language work, without a compiler's size and sluggish code.

Best of all, every line of the toolkit's sorce code is there for you to read, and every module's operation is explained with the clarity and good humor for which Dave Cortesi's writing is known.





Order the Z80 Software on Disk! Save Yourself the Frustration of File Entry

All the software in **Dr. Dobb's Z80 Toolbook**—the programs plus the entire toolkit, both as source code and as object modules for both CP/M 2.2 and CP/M Plus—is yours on disk. (A Z80 microprocessor and a Digital Research International RMAC assembler or equivalent are required.)

Receive *Dr. Dobb's Toolbook for Z80,* along with the software on disk, together for only $40!

Dr. Dobb's Toolbook for Z80
Item #022 $25

Dr. Dobb's Toolbook for Z80, together with software on disk. Please specify one of the following formats: 8″ SS/SD; Apple; Osborne: Kaypro.

Item #022A $40

Most of the programs are included in the book; however, the disk is necessary for complete listings

To Order:

To order any of *Dr. Dobb's* products, return the order form at the end of this catalog, or

CALL TOLL-FREE
1-800-528-6050 EXT. 4001

and refer to product item number, title, and disk format.

For customer service questions,
CALL M&T PUBLISHING, INC.
415-366-3600 EXT. 216

M&T BOOKS

Dr.Dobb's Catalog Order Form

ORDER NOW

NAME ______________________________

(Please use street address, not P.O.Box)

ADDRESS ______________________________

CITY ______________ STATE ________ ZIP ________

DAY PHONE ______________________________

For disk orders, please indicate format. Refer to ad for standard format availability for each product. Special formats available for additional $10 each.

☐ MS/DOS ☐ Macintosh ☐ CP/M

_____ Zenith Z-100 DS/DD

_____ Apple

_____ Osborne

_____ Kaypro

_____ 8″ SS/SD

For Faster Service On Credit Card Orders

CALL TOLL FREE 1-800-528-6050 Ext. 4001

Please Refer To Item # When Ordering

Or, Fill Out This Postage Paid, Self-Mailing Order Form And Return To:

M&T PUBLISHING INC.
501 GALVESTON DRIVE
REDWOOD CITY, CA 94063

QUANTITY	ITEM#	DESCRIPTION	UNIT PRICE	TOTAL PRICE

SUB-TOTAL ________

SALES TAX ________

SHIPPING ________

TOTAL ORDER ________

CA residents must add applicable sales tax on merchandise total ________ %
(CA residents must add sales tax to all items EXCEPT *Dr. Dobb's Sourcebook* #004)
Shipping must be included with order. See rates below.

In U.S. For Bound Volumes, add $2.25 per book. Add $8.75 for special C Packages. For other books and disks, add $1.75 per item.
Outside U.S. For Bound Volumes, add $5.25 per book surface mail. Add $18 surface mail for Special C Packages. For other books and disks, add $3.25 per item surface mail. Foreign airmail rates available on request.

☐ VISA

☐ MASTERCARD

☐ AMERICAN EXPRESS

☐ CHECK
(make checks payable to M&T Publishing)

NAME ON CARD ______________________________

ACCOUNT NO. ______________________________

EXPIRATION DATE ______________________________

SIGNATURE ______________________________

PROMPT DELIVERY! DEALER INQUIRY WELCOME!

M&T BOOKS

3117S

Dr. Dobb's Catalog Order
Please Rush!

Please fold along fold-line and staple or tape closed.

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

First Class   Permit No. 790   Redwood City, CA

Postage Will Be Paid By Addressee

Dr. Dobb's Catalog
501 Galveston Dr.
Redwood City, CA 94063

Please fold along fold-line and staple or tape closed.

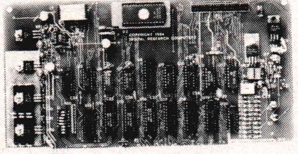
ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

S100 EPROM PROGRAMMER

OUR NEWEST DESIGN, FOR FAST EFFICIENT PROGRAMMING OF THE MOST POPULAR EPROM'S ON YOUR S100 MACHINE. COMES WITH MENU DRIVEN SOFTWARE THAT RUNS UNDER CP/M 2.2 (8 INCH). PC BOARD SET CONSISTS OF (S100) MAIN LOGIC BOARD REMOTE PROGRAMMING CARD AND SIX PERSONALITY MINI BOARDS FOR 2716, 2532, 2732, 2732A, 2764, AND 27128. SOLD AS BARE PC BOARD SET ONLY WITH FULL DOC. SOFTWARE FEATURES "FAST" PROGRAMMING ALGORITHM. FOR Z80 BASED SYSTEMS.



PC BOARD SET, FULL DOCUMENTATION, 8 IN. DISKETTE WITH SOFTWARE.

NEW! $69.95

128K S100 STATIC RAM/EPROM BOARD

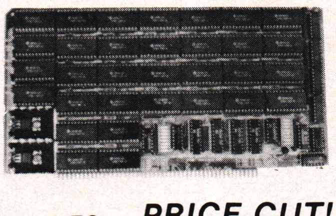
JUST OUT! USES POPULAR 8K X 8 STATIC RAMS (6264) OR 2764 EPROMS. FOR 8 OR 16 BIT DATA TRANSFERS! IEEE 696 STANDARD. LOW POWER. KITS ARE FULLY SOCKETED. FULL DOC AND SCHEMATICS INCLUDED. 24 BIT ADDRESSING.

NEW!

$59.95 BARE PC BOARD — $219.00 128K RAM KIT — $139.00 128 EPROM KIT

64K S100 STATIC RAM

$99.00 KIT



LOW POWER!

150 NS ADD $10

BLANK PC BOARD WITH DOCUMENTATION $49.95

SUPPORT ICs + CAPS $17.50

FULL SOCKET SET $14.50

FULLY SUPPORTS THE NEW IEEE 696 S100 STANDARD (AS PROPOSED)

ASSEMBLED AND TESTED ADD $50

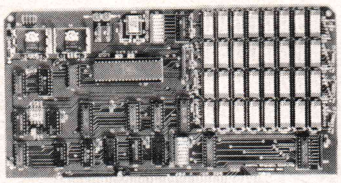
FEATURES: *PRICE CUT!*

* Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
* Fully supports IEEE 696 24 BIT Extended Addressing.
* 64K draws only approximately 500 MA.
* 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
* SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
* 2716 EPROMs may be installed in any of top 48K.
* Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
* Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
* BOARD may be partially populated as 56K.

256K S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



FEATURES:

* 256K on board, using + 5V 64K DRAMS.
* Uses new Intel 8203-1 LSI Memory Controller.
* Requires only 4 Dip Switch Selectable I/O Ports.
* Runs on 8080 or Z80 S100 machines.
* Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
* Provisions for Battery back-up.
* Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
* The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
* Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB (WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE) $49.95 (8203-1 INTEL $29.95)

(ADD $50 FOR A&T) #LS-100

$129.00 (FULL 256K KIT)

PANASONIC

Green Screen - Video Monitors

25 MHZ. TYPICAL BANDWIDTH!!!

Brand New In The Box! 9-Inch Screen

#K-904B1 (Chassis #Y08A) Open Frame Style

$29.95 EA.

GROUP SPECIAL: 4 for $99.00 WITH DATA & SCHEMATIC

(USA SHIPPING: $3. PER UNIT. CANADA: $7. PER UNIT)

COMPUTER MANUFACTURER'S EXCESS. STILL IN ORIGINAL PANASONIC BOXES. THE CRT TUBE ALONE WOULD COST MORE THAN OUR PRICE FOR THE COMPLETE UNIT. FOR SPLIT VIDEO (TTL INPUTS) OPERATION, NOT COMPOSITE VIDEO. OPERATES FROM 12VDC AT 1 AMP. VERTICAL INPUT IS 49 TO 61 HZ. HORIZONTAL INPUT: 15,750 HZ ± 500 HZ. RESOLUTION IS 800 LINES AT CENTER 650 LINES AT CORNERS.

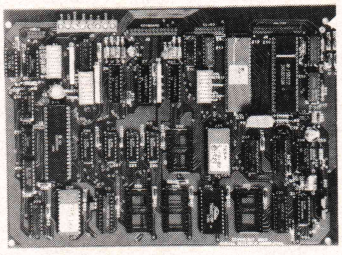
ZRT-80
CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:

* Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
* RS232 at 16 BAUD Rates from 75 to 19,200.
* 24 x 80 standard format (60 Hz).
* Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
* Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
* Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
* 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
* Composite or Split Video.
* Any polarity of video or sync.
* Inverse Video Capability.
* Small Size: 6.5 x 9 inches.
* Upper & lower case with descenders.
* 7 x 9 Character Matrix.
* Requires Par. ASCII keyboard.

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD $10



$89.95 #ZRT-80 — A&T ADD $50

(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716 CHAR. ROM. 2732 MON. ROM $49.95

SOURCE DISKETTE - ADD $10

SET OF 2 CRYSTALS - ADD $7.50

THE NEW 65/9028 VT
ANSI VIDEO TERMINAL BOARD!

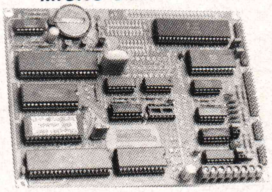
★ FROM LINGER ENTERPRISES ★

A second generation, low cost, high performance, mini sized, single board for making your own RS232 Video Terminal. Use as a computer console or with a MODEM for hook up to any of the telephone-line computer services.

FEATURES:

* Uses the new SMC 9028 Video Controller Chip coupled with a 6502A CPU.
* RS-232 at 16 Baud Rates from 50 to 19,200
* On board printer port!
* 24 X 80 format (50/60 Hz).
* For 15,750 Hz (Horiz.) monitors.
* 3 Terminal Modes: H-19, ADM3A, and ANSI X 3.64-1979
* Wide and thin-line graphics.
* White characters on black background or reversed.
* Character Attributes: De-Inten, Inverse, Underline and Blank.
* Low Power: 5VDC @ .7A, ± 12VDC @ 20MA.
* Mini size: 6.5 X 5 inches.
* Composite or split video.
* 5 X 8 Dot Matrix characters (U/L case) with descenders.
* Answer back capability.
* Battery backed up status memory.
* For ASCII parallel keyboard.

MICRO SIZE!



$99.95 (Full Kit)

ADD $40 FOR A&T

SOURCE DISKETTE: PC/XT FORMAT 5¼ IN. $15

OPTIONAL EPROM FOR PC/XT STYLE SERIAL KEYBOARD: $15

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

TERMS: Add $3.00 postage. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance.

*TM OF DIGITAL RESEARCH INC. (CALIF.)

WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

Listing One (Text begins on page 30.)

```
B:PAPERS.BLK

Screen   12*
\ FORTH-83 Standard type control structures            GWS 86Mar31
\ note: !0 is 0 SWAP !    ADDR, is ,    (S is (
: >MARK   HERE 2 ALLOT   ;          (S - a )( mark forward branch )
: >RESOLVE   HERE SWAP !   ;      (S a - )( patch forward branch )
: <MARK   HERE   ;          (S - a )( destination for back branch )
: <RESOLVE   ADDR,   ;          (S a - )( compile reference to a )
VARIABLE LEAVE-LIST
: >MARKLIST        (S a - )( extend list at a, link in dictionary )
   HERE SWAP DUP @ ADDR, ( link) ! ( new head) ;
: >RESOLVESLIST          (S a - )( resolve all nodes in a to here )
   DUP @ BEGIN   DUP WHILE   DUP @ SWAP >RESOLVE   REPEAT
   DROP !0   ;    1 2 +THRU

Screen   13
\   conditional compilers - if/else/then begin/while GWS 86Mar31
: IF              (S - a / f - )( compile to branch if f is false )
   COMPILE ?BRANCH   >MARK   ; IMMEDIATE
: ELSE             (S a1 - a2 / - )( compile alternate to IF clause )
   COMPILE BRANCH   >MARK   SWAP >RESOLVE   ; IMMEDIATE
: THEN             (S a - / - )( resolve latest forward reference )
   >RESOLVE   ; IMMEDIATE
: BEGIN   <MARK   ; IMMEDIATE    (S - a / - )( mark loop start )
: WHILE   [COMPILE] IF   ; IMMEDIATE (S - a / f - )( loop exit )
: REPEAT          (S a1 a2 - / - )( branch to beginning of loop )
   COMPILE BRANCH   SWAP <RESOLVE   >RESOLVE   ; IMMEDIATE
: UNTIL (S a - / f - )( branch to beginning of loop until true )
   COMPILE ?BRANCH   <RESOLVE   ; IMMEDIATE

Screen   14
\   do loops                                              GWS 86Mar31
: LEAVE                    (S - / - )( compile exit from structure )
   COMPILE (LEAVE)   LEAVE-LIST >MARKLIST   ; IMMEDIATE
: DO                   (S - n a / n1 n2 - )( initiate counted loop )
   COMPILE (DO)   LEAVE-LIST @   LEAVE-LIST !0   <MARK   ;
   IMMEDIATE
: LOOP                 (S n a - / - )( compile increment loop end )
   COMPILE (LOOP)   <RESOLVE   LEAVE-LIST >RESOLVESLIST
   LEAVE-LIST !   ; IMMEDIATE
: +LOOP           (S n a - / u - )( compile u incremented loop end )
   COMPILE (+LOOP)   <RESOLVE   LEAVE-LIST >RESOLVESLIST
   LEAVE-LIST !   ; IMMEDIATE
```

End Listing One

Listing Two

```
B:PAPERS.BLK

Screen   15*
\   typical BEGIN loop extensions                         GWS 86Mar31
: RESOLVES      (S 0..a - )( resolve forward branches a until 0 )
   BEGIN   ?DUP WHILE   >RESOLVE   REPEAT   ;
: BEGIN   0 <MARK   ; IMMEDIATE     (S 0 a - )( mark loop start )
: WHILE            (S a1 - a2 a1 / f - )( conditional loop exit )
   [COMPILE] IF   SWAP   ; IMMEDIATE
: REPEAT           (S 0..an a - / - )( terminate repeating loop )
   COMPILE BRANCH <RESOLVE   RESOLVES   ; IMMEDIATE
: UNTIL          (S 0..an a - / f - )( terminate conditional loop )
   COMPILE ?BRANCH <RESOLVE   RESOLVES   ; IMMEDIATE
```

End Listing Two

Listing Three

```
B:PAPERS.BLK

Screen   3*
\ Proposed Standard Control Structures                 GWS 86Mar31
\ note: !0 is 0 SWAP !    ADDR, is ,    (S is (
: >MARK   HERE 2 ALLOT   ;          (S - a )( mark forward branch )
: >RESOLVE   HERE SWAP !   ;      (S a - )( patch forward branch )
: <MARK   HERE   ;          (S - a )( destination for back branch )
: <RESOLVE   ADDR,   ;          (S a - )( compile reference to a )
: >MARKLIST        (S a - )( extend list at a, link in dictionary )
   HERE SWAP DUP @ ADDR, ( link) ! ( new head) ;
: >RESOLVELIST          (S a - )( resolve top node in a to here )
   DUP @ DUP @ ROT ! ( unlink top node) >RESOLVE   ;
: >RESOLVESLIST          (S a - )( resolve all nodes in a to here )
   DUP @ BEGIN   DUP WHILE   DUP @ SWAP >RESOLVE   REPEAT
   DROP !0   ;    1 6 +THRU

Screen   4
\   compilation list initialization                    GWS 86Mar31
ORPHAN                                       ( make headless words )
VARIABLE IF-LIST   VARIABLE LEAVES-LIST   VARIABLE LEAVE-LIST
VARIABLE LEAVE-CF
: INIT-LISTS                    (S - )( reset all list pointers )
   IF-LIST !0   LEAVE-LIST !0   LEAVES-LIST !0   ;
: SAVE-LISTS          (S - x x x x )( save current list pointers )
   LEAVE-CF @   IF-LIST @   LEAVE-LIST @   LEAVES-LIST @
   INIT-LISTS   ;
: RESTORE-LISTS  (S - x x x x )( restore current list pointers )
   ( could check here for unresolved structures)
   LEAVES-LIST !   LEAVE-LIST !   IF-LIST !   LEAVE-CF !   ;
ADOPT                                          ( make headed words )

Screen   5
\ Conditional compilers - if/else/then & case          GWS 86Mar31
: IF                  (S - / f - )( compile to branch if f is false )
   COMPILE ?BRANCH   IF-LIST >MARKLIST   ; IMMEDIATE
: ELSE                (S - / - )( compile alternate to IF clause )
   COMPILE BRANCH   IF-LIST >MARKLIST   IF-LIST @ ( if branch)
   >RESOLVELIST   ; IMMEDIATE
: THEN                (S - / - )( resolve latest forward reference )
   IF-LIST >RESOLVELIST   ; IMMEDIATE
: CASE            (S - x x x x / ? - ? )( setup for case statement )
   SAVE-LISTS   ['] BRANCH LEAVE-CF !   ; IMMEDIATE
: ENDCASE     (S - / x x x x - )( restore lists, resolve leaves )
   LEAVES-LIST >RESOLVESLIST   RESTORE-LISTS   ; IMMEDIATE

B:PAPERS.BLK

Screen   6*
\   common loop end and exit                           GWS 86Mar31
ORPHAN                                       ( make headless words )
: LOOPEND        (S x x x x a1 a2 - )( resolve list a2 & branch )
( a1, restore values x, transfer leaves-list to if-list )
   SWAP <RESOLVE ( back branch) >RESOLVESLIST ( forward branch)
   LEAVES-LIST @ ?DUP IF   >R   RESTORE-LISTS   IF-LIST @   R@
   BEGIN   DUP @ WHILE   @   REPEAT ( find leaves list end) !
   ( link to if list) R> IF-LIST !   ELSE   RESTORE-LISTS
   THEN   ;
ADOPT                                          ( make headed words )
: OUTSIDE   (S - / - )( allow LEAVES outside current loop level)
   IF-LIST @ DUP @ IF-LIST ! ( unlink) LEAVES-LIST @ OVER !
   COMPILE-UNNEST   LEAVES-LIST ! ( relink) ; IMMEDIATE

Screen   7
\   do loops                                              GWS 86Mar31
: LEAVE                    (S - / - )( compile exit from structure )
   LEAVE-CF @ ADDR,   LEAVE-LIST >MARKLIST   ; IMMEDIATE
: LEAVES              (S - / - )( compile exit to outside structure )
   LEAVE-CF @ ADDR,   LEAVES-LIST >MARKLIST   [COMPILE] THEN ;
   IMMEDIATE
: DO            (S - x x x x a / u u - )( initiate counted loop )
   SAVE-LISTS   ['] (LEAVE) LEAVE-CF !   COMPILE (DO)
   <MARK   ; IMMEDIATE
: LOOP            (S x x x x a - / - )( compile increment loop end )
   COMPILE (LOOP)   LEAVE-LIST LOOPEND   ; IMMEDIATE
: +LOOP               (S x x x x a - / u - )( compile u+ loop end )
   COMPILE (+LOOP)   LEAVE-LIST LOOPEND   ; IMMEDIATE

Screen   8
\   more loops                                            GWS 86Mar31
: BEGIN              (S - x x x x a / - )( mark start of a loop )
   [COMPILE] CASE   <MARK   ; IMMEDIATE
: REPEAT          (S x x x x a - / - )( terminate repeating loop )
   COMPILE BRANCH   IF-LIST LOOPEND
   LEAVE-LIST >RESOLVESLIST   ; IMMEDIATE
: UNTIL           (S x x x x a - / - )( terminate repeating loop )
   COMPILE ?BRANCH   IF-LIST LOOPEND
   LEAVE-LIST >RESOLVESLIST   ; IMMEDIATE
: WHILE   [COMPILE] IF   ; IMMEDIATE   (S - )( for compatibility)
```

End Listing Three

Listing Four

```
B:PAPERS.BLK

Screen    9*
\   suggested extensions                                GWS 86Mar31

: ?LEAVE                    (S - / f - )( leave do loop if tf )
  COMPILE (?LEAVE)    LEAVE-LIST >MARKLIST    ; IMMEDIATE
: ?LEAVES                   (S - / f - )( leave do loop if tf )
  COMPILE (?LEAVE)    LEAVES-LIST >MARKLIST    ; IMMEDIATE

: THENS                (S - / - )(  resolve all outstanding IFs )
  IF-LIST >RESOLVESLIST    ; IMMEDIATE
: ELSES   (S - / - )( resolve all outstanding IFs w/common ELSE )
  [COMPILE] ELSE    IF-LIST @ >RESOLVESLIST    ; IMMEDIATE
```

End Listing Four

Listing Five

```
Previously Proposed Solutions               Proposed Solution

BEGIN ...                                   same
   WHILE ...
   WHILE ...
   ...
REPEAT

BEGIN ...                                   same
   WHILE ...
   WHILE ...
   ...
UNTIL

BEGIN ...                                   BEGIN ...
   WHILE ...                                   WHILE
   ANDWHILE ...                                WHILE
   ANDWHILE ...                                WHILE
   ...                                         ...
REPEAT                                      REPEAT

BEGIN ...                                   BEGIN ...
   WHILE   aa                                  NOT IF    ff    LEAVES   aa
   WHILE   bb                                  NOT IF    ee    LEAVES   bb
   WHILE   cc                                  WHILE   cc
   ...                                         ...
REPEAT   dd                                 REPEAT   dd
   <WHILE   ee
   <WHILE   ff
<END                                        THEN   THEN

BEGIN ...                                   see below
   IF ... LEAVE THEN
   IF ... LEAVE THEN
   ...
REPEAT

BEGIN ...                                   BEGIN ...
   UNLESS ... FINISH                           IF ... LEAVES
   UNLESS ... FINISH                           IF ... LEAVES
   ...
AGAIN                                       REPEAT THEN THEN

DO ...                                      DO ...
   PERHAPS ... ESCAPE                          IF ... LEAVES
   PERHAPS ... ESCAPE                          IF ... LEAVES
   ...                                         ...
LOOP ...                                    LOOP ...
ESCAPED ...                                 THEN THEN ... (or ELSE ...
                                                                  THEN)

DO ...                                      DO ...
   IF ... LEAVE THEN    aa                     IF ... LEAVES     aa
LOOP--FALLTHRU:    bb                       LOOP   bb
THEN    cc                                  THEN   cc

DO ...                                      DO ...
   WHEN ...                                    NOT IF    LEAVE THEN ...
LOOP                                        LOOP

DO ...                                      DO ...
   NOTWHEN ...                                 IF    LEAVE THEN ...
LOOP                                        LOOP

DO ...                                      DO ...
   IF   LEAVE THEN ...                         IF    LEAVES
EXITING LOOP                                LOOP
...                                         ...
THEN                                        THEN

none                                        DO ...
                                               DO ...
                                                  IF    LEAVES
                                                  ...
                                               LOOP OUTSIDE
                                            LOOP
                                            ...
                                            THEN

none                                        BEGIN ...
                                               BEGIN ...
                                                  IF   LEAVES
                                                  ...
                                               REPEAT OUTSIDE
                                            REPEAT
                                            ...
                                            THEN

<STEPS ...                                  CASE
   &IF ...                                     IF ...
   &IF ...                                     IF ...
STEPS>                                         THENS
                                            ENDCASE

IF ... ELSE ...                             IF ... ELSE ...
THENIF ... ELSE                             IF ... ELSE ...
THENIF ... ELSE                             IF ... ELSE ...
...                                         ...
THEN                                        THENS   or

                                            CASE
                                               IF ... LEAVES
                                               IF ... LEAVES
                                               ...
                                            ENDCASE

IF ...                                      IF ...
   ANDIF ...                                   IF ...
      ANDIF ...                                   IF ...
      ...                                         ...
( ELSE)                                     ( ELSES)
THEN                                        THENS ( THEN)

CASE ...                                    CASE ...
   OF ... ENDOF                                OVER = IF ... LEAVES
   OF ... ENDOF                                OVER = IF ... LEAVES
   ...                                         ...
ENDCASE                                     DROP ENDCASE
```

End Listings

/* RELAX PROGRAMMERS */

We know how frustrating programming can be . . . and we decided to do something about it.

We have created an Environment — just for programmers (you know who you are), that relieves some of the frustration of programming and increases your productivity.

We called it **KeyFire. KeyFire** is an Integrating Programming Environment that combines YOUR editor, compiler, linker, debugger, and make utility into a single entity.

- Reduces the steps of the development cycle (think, edit, compile, link, debug) to single key stroke operations.
- Maintains an Environment file of the parameters for each step.
- Switch easily from Environment to Environment, so you can work on several programs at once — all without leaving **KeyFire.**
- Perform multiple compiles and link in one step.
- Define macros of DOS command sequences.
- Built-in Help utility filled with lots of often needed information (including DOS commands and their options) and you can add more information yourself.
- We know it's great; we know you'll like it, because we constructed **KeyFire** with **KeyFire!**

$77.00 USA dollars

Ideal for De Smet C. Lattice C and most one or two pass compilers regardless of the language.

For IBM/XT/AT and compatibles, DOS 2.1 or later.

Send check or money order to:

Software Construction
467 Saratoga Ave. #256
San Jose, CA 95129

California residents add sales tax. Foreign orders add $5.
De Smet C is a trademark of Cware Corp.
Lattice C is a trademark of Lattice Inc.

Circle **no. 177** on reader service card.

Listing One (Text begins on page 40.)

```
****************************************************************
*     RAFOS FORTH V1.0        26 March 1986                     *
*     -- ROM #1 of 2                                             *
*                                                                *
*   (C) Copyright 1986, Everett Carter. All rights reserved.    *
*                                                                *
*   This FORTH is a subset of the FORTH-79 standard.            *
*   Some changes have been made in order to save on             *
*   space in the limited memory of the float.                   *
*                                                                *
****************************************************************
*
*     EQUATES FOR ROM 2
QUES     EQU    $1000
*
TOG6     EQU    $105F            TOGGLE
IMM9     EQU    $1089            IMMEDIATE
DFND     EQU    $109B            -FIND
COU5     EQU    $10B1            COUNT
ZERO     EQU    $10C3            0
ONE      EQU    $10D8            1
TWO      EQU    $10EF            2
TWOP     EQU    $1106            2+
LBRAK    EQU    $1121            [
RBRAK    EQU    $1131            ]
DEFS     EQU    $1143            DEFINITIONS
PLUS     EQU    $1155            +
MINUS    EQU    $117C            -
UMULT    EQU    $11A7            U*
*
PAD      EQU    $1289            PAD
LSHP     EQU    $129B            <#
OVER     EQU    $12AB            OVER
SPGR     EQU    $12CE            #>
TOR      EQU    $12E6            >R
RTO      EQU    $130A            R>
RFTCH    EQU    $132E            R@
ROT      EQU    $134F            ROT
HOLD     EQU    $1361            HOLD
*
COMP     EQU    $1557            COMPILE
SEMI     EQU    $156D            ;
COLON    EQU    $157F            :
TICK     EQU    $159B            '
VAR8     EQU    $15C5            VARIABLE
*
I1       EQU    $17EE            I
*
LATEST EQU I1-6                  Last Dictionary entry
*
ROM      EQU    $1800            ROM #1 start address
*
CR       EQU    $0D              CARRIAGE RETURN
LF       EQU    $0A              LINE FEED
BL       EQU    $20              BLANK
BS       EQU    $08              Back Space
DEL      EQU    $7F              Delete
*
DDR      EQU    4                DATA DIRECTION REGISTER OFFSET
*
PORTA    EQU    0                I/O PORT 0
PORTB    EQU    1                I/O PORT 1
PUT      EQU    PORTB            SERIAL I/O PORT
*
INITSP   EQU    $7F              INITIAL STACK POINTER VALUE
STACK    EQU    INITSP-5         TOP OF STACK
MEMSIZ   EQU    $2000            MEMORY ADDRESS SPACE SIZE
*
SP0      EQU    $0F00
RP0      EQU    $0E00
TIB      EQU    $0D80
*
*
*
*    RAM VARIABLES
*
*        ORG    $10              ON-CHIP RAM (112 BYTES)
*
ATEMP    EQU    $10              TEMP USED IN PUTDEC
XTEMP    EQU    $11              INDEX TEMPORARY
GETR     EQU    $12              PICK & DROP TEMPORARY
COUNT    EQU    $16              NUMBER OF BITS LEFT TO get/send
CHAR     EQU    $17              Current input/output character
*
BYTCNT   EQU    $1E              bytcnt.
WTIME    EQU    $20              TIMER INTERRUPT FROM WAIT STATE
*
*
PH       EQU $21             MISC SCRATCH AREAS
PL       EQU $22
TEMPA    EQU $23
TEMPB    EQU $24
QH       EQU $25
QL       EQU $26
TEMP     EQU $27
TERM     EQU $28
*
         ORG $0029
*
IN       FCB #0              Where FORTH will look for input
OUT      FCB #0
COUNTR   FCB #0
DP       FDB #$01D0          The initial Dictionary pointer
START    FDB #0              The start up vector
*
*
IP       FDB #0              THE FORTH INSTRUCTION POINTER
RP       FCB #0              THE RETURN POINTER OFFSET
SP       FCB #0              THE STACK POINTER OFFSET
BASE     FCB #$10
*
USER     EQU *               The space for USER variables
FENCE    EQU 0               USER + 0
         FDB #0                       INITIALIZE USER VARS
STATE    EQU 2               USER + 2
         FDB #0
FORTH    EQU 4               USER + 4
         FDB #0
CONTEXT  EQU 6               USER + 6
         FDB USER+FORTH
CURRENT  EQU 8               USER + 8
         FDB USER+FORTH
HLD      EQU $0A             USER + $0A
         FDB #0
*
      ORG $0080
*
*        The start of the INNER interpreter
*
DOCOL  LDX RP                          * Push IP to RS
*
DOCOL1 EQU DOCOL
*
       DECX
       LDA IP+1
       STA RP0,X
       DECX
       LDA IP
       STA RP0,X
       STX RP
       LDA NEXT1+2
       ADD #2
       STA IP+1
       LDA NEXT1+1
       ADC #0
       STA IP
*
*      fall thru to NEXT
*
NEXT   LDA IP+1                NEXT The Inner Interpreter
       STA CA+2                SELF-MODIFYING
       LDA IP
       STA CA+1
CA     LDA SP0               -- SP0 is a dummy
       STA NEXT1+1
       LDA IP+1
       ADD #1
       STA CA2+2
       LDA IP
       ADC #0
       STA CA2+1
CA2    LDA SP0                 -- SP0 is a dummy
       STA NEXT1+2
       LDA IP+1
       ADD #2
       STA IP+1
       LDA IP
       ADC #0
       STA IP
NEXT1  JMP COLD                -- COLD is a dummy
*
*       SELF MODIFYING CODE FIRST
*
LOAD     STA SP0,X                      STA (HERE),X
         RTS                            move A to HERE+X
*
GET      LDA SP0,X                      LDA (HERE),X
         RTS                            get HERE+X into A
*
       FCB 4                  TYPE  -- SELF MODIFYING
       FCC 'TYP'
       FDB #0                 end link
TYPE   LDX SP
       INCX                   Drop high byte
       LDA SP0,X
       INCX
       STA COUNTR             COUNTR = byte count
       LDA SP0,X
       INCX
       STA TYSCR+1
       LDA SP0,X
       INCX
       STA TYSCR+2
       STX SP
       CLR OUT
```

```
        TST COUNTR
        BEQ TXIT
TLOOP   LDX OUT
TYSCR   LDA SP0,X            -- SP0 is a dummy
        JSR OUTCHAR
        INC OUT
        LDA OUT
        SUB COUNTR
        BMI TLOOP
TXIT    JMP NEXT
*
*
        FCB 6                <FIND> -- SELF MODIFYING
        FCC '<FI'
        FDB TYPE-6           link to TYPE
FIN6    LDX SP
        LDA SP0,X            get addr1 high
        INCX
        STA GET+1
        LDA SP0,X            addr1 low
        INCX
        STA GET+2
        LDA SP0,X            get addr2 high
        INCX
        STA FINSCR+1
        STA FINCNT+1
        LDA SP0,X
        INCX
        STA FINSCR+2
        STA FINCNT+2
        STX SP
FINCNT  LDA SP0              -- SP0 is a dummy
        STA COUNTR           save byte count
        TSTA                 count = 0 ?
        BEQ NONE
FINLP1  CLRX
FINLP2  JSR GET
        AND #$7F             ignore bit 7
FINSCR  CMP SP0,X            -- SP0 is a dummy
        BNE NFND
        CPX #3               X = 3 ? if so quit as FOUND
        BEQ FOUND
        CPX COUNTR           X = count ?
        BEQ FOUND
        INCX
        BRA FINLP2
NFND    LDX #4               Not found, go to next element
        JSR GET
        STA ATEMP
        INCX
        JSR GET
        ORA ATEMP            =0 ?
        BEQ NONE             if yes, end of list
        JSR GET              else move new pointer to get
        STA GET+2
        LDA ATEMP
        STA GET+1
        BRA FINLP1           and try again
NONE    LDX SP               nothing, push a FALSE to stack
        CLRA
        BRA FQUIT
FOUND   LDX SP
        LDA GET+2            push CA of found word
        ADD #6
        DECX
        STA SP0,X
        LDA GET+1
        ADC #0
        DECX
        STA SP0,X
        STX SP
        CLRX
        JSR GET              get the byte count and push it
        LDX SP
        DECX
        STA SP0,X
        CLRA
        DECX
        STA SP0,X
        LDA #$FF             push a TRUE flag
FQUIT   DECX
        STA SP0,X
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 4                BRAN -- SELF MODIFYING
        FCC 'BRA'
        FDB  FIN6-6          link to <FIND>
BRAN    LDA IP
        STA BRSC1+1
        STA BRSC2+1
        LDA IP+1
        STA BRSC1+2
        STA BRSC2+2
        LDX #1
BRSC1   LDA SP0,X
        ADD IP+1
        STA IP+1
        CLRX
BRSC2   LDA SP0,X
        ADC IP
```

(continued on next page)

AT LAST: Professional Typesetting Capability For PC Users

With **PC TEX**™ — the best-selling full implementation of Professor Don Knuth's revolutionary typesetting program TEX.

FINEST Typeset Quality Printing From:
dot matrix   laser   phototypesetter

$$\sum_{i=1}^{\infty}\frac{1}{i} \qquad \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \qquad \int_{-\infty}^{\infty} e^{-x^2}\,dx$$

WIDEST Range Of Output Device Drivers:

- Epson FX, LQ
- Toshiba
- Corona LP-300*
- HP LaserJet*
- Apple LaserWriter
- APS-5 phototypesetter
- Screen preview, with EGA or Hercules card

MOST COMPLETE Product Offering:

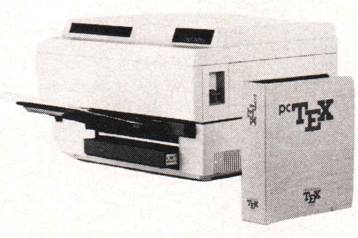
PC TEX (not copy protected) includes the following:

- Our specially written *PC TEX Manual*, which enables you to start using TEX right away.
- Custom "macro packages" that provide formats for letters, manuals, technical documents, etc.
- The LATEX document preparation system, a full-featured macro package for preparing articles, books, reports, etc., and LATEX User's Manual.
- AMS-TEX, developed by the *Amer. Math. Society* for professional mathematical typesetting.

Site licenses, volume discounts, and interfaces to PC Paintbrush, PC Palette, FancyFont and Fontrix are also available.

PRICED FROM ONLY $249.00!

(Printer drivers and interfaces additional.)



Laser printer, fonts & software from $2995.00

For IBM PC/XT, AT or compatible, DOS 2.0 or higher, and 512K RAM. Hard disk required for printer drivers and fonts.
*HP LaserJet and Corona require additional interface boards.

For more information call or write:

Personal TEX, Inc.

20 Sunnyside, Suite H, Mill Valley, CA 94941 (415) 388-8853

This ad, with space for the photograph, produced by PC TEX. Typeset on the Epson FX80, the Corona LP-300 laser printer, and the Autologic APS-5 phototypesetter.

TEX is a trademark of the American Mathematical Society. Manufacturers' product names are trademarks of individual manufacturers.

Circle **no. 76** on reader service card.

Listing One (Listing continued, text begins on page 40.)

```
        STA IP
        JMP NEXT
*****************************************************************
*
*      NO SELF MODIFYING CODE BEYOND THIS POINT
*
OFFSET EQU *
        ORG ROM+OFFSET          *  ROM #2 ORIGIN
*
*
        FCB 7                  0BRANCH
        FCC '0BR'
        FDB BRAN-6             link to BRAN
ZBRAN   LDX SP
        LDA SP0,X
        INCX
        ORA SP0,X
        INCX
        STX SP
        TSTA
        BNE ZBREX
        JMP BRAN
ZBREX   LDA IP+1               bump IP past offset
        ADD #2
        STA IP+1
        LDA IP
        ADC #0
        STA IP
        JMP NEXT
*
        FCB 4                     EXIT
        FCC 'EXI'
        FDB ZBRAN-6                link to 0BRANCH
EXIT    LDX RP                    Pop RS into IP
        LDA RP0,X                 High byte
        INCX
        STA IP
        LDA RP0,X                 then low byte
        INCX
        STA IP+1
        STX RP
        JMP NEXT
*
        FCB 7                     EXECUTE
        FCC 'EXE'
        FDB EXIT-6                link to EXIT
EXE7    LDX SP                    Pop SP into W (NEXT1+1)
        LDA SP0,X                 First high byte
        INCX
        STA NEXT1+1
        LDA SP0,X              Then low byte
        INCX
        STA NEXT1+2
        STX SP
        JMP NEXT1
*
INLINE JSR CRLF
        LDA #BL
        CLR COUNTR
        CLRX                    Clear line buffer
INLP1   STA TIB,X
        INCX
        CPX #$7E                Buffer end ?
        BNE INLP1
        CLR IN
        CLRX                    Clear buffer pointer
INLP2   JSR GETCHAR              ( X = IN )
        CMP #DEL                = DELETE ?
        BNE INTST2              branch if not
INDEL   CPX #0
        BEQ INLP2               Skip if IN (LBP) = 0 already
        DECX
        LDA #BL                 DELETE CHAR
        STA TIB,X
        LDA #BS
        JSR OUTCHAR
        LDA #BL
        JSR OUTCHAR
        LDA #BS
        JSR OUTCHAR
        BRA INLP2
INTST2 CMP #BS                 maybe its a backspace
        BEQ INDEL
        CMP #CR                 or a CR
        BEQ INEX
        STA TIB,X
        CPX #$7D
        BHS INSKP
        INCX
INSKP   JSR OUTCHAR
        BRA INLP2               Back to main loop
INEX    LDA #BL
        JSR OUTCHAR
        JMP NEXT
*
        FCB 4                     EMIT
        FCC 'EMI'
        FDB EXE7-6                link to EXECUTE
EMIT    LDX SP
        INCX                      drop high byte
        LDA SP0,X
        INCX
        STX SP
        JSR OUTCHAR
        JMP NEXT
*
        FCB 2                    BL
        FCC 'BL '
        FDB EMIT-6               link to EMIT
BL2     LDX SP
        LDA #BL
        DECX
        STA SP0,X
        CLRA
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 4                    WORD
        FCC 'WOR'
        FDB BL2-6                link to BL
WORD    LDA DP                   start by setting up LOAD
        STA LOAD+1
        LDA DP+1
        STA LOAD+2
        CLR COUNTR
        CLRA
        CLRX
        JSR LOAD               clear DP
        LDX SP                 get terminator
        INCX                   drop high byte
        LDA SP0,X
        STA TERM
        INCX
        STX SP
        LDX IN                 get INput pointer
        CMP #BL                seperator = space ?
        BNE TOK
IGNBL   LDA TIB,X              ignore blank
        CMP #BL
        BNE TOK
        INCX
        BRA IGNBL
TOK     INC COUNTR
        LDA TIB,X
        STX XTEMP               save X
        LDX COUNTR
        JSR LOAD               move char to DP + COUNTR
        LDX XTEMP               get X back
        INCX
        CMP #$80               character = buffer end ?
        BEQ WORXIT
        CMP TERM
        BNE TOK                continue unless terminator
WORXIT STX IN                  save new value of IN
        LDA COUNTR             move count-1 to DP
        DECA
        CLRX
        JSR LOAD
        LDA DP+1               Push DP addr to stack
        LDX SP
        DECX
        STA SP0,X
        LDA DP
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
MPY16   LDX #$10               16 bit X 16 bit multiply   32 bit result
        CLR TEMPA
        CLR TEMPB
        ROR QH
        ROR QL
MPYNXT BCC ROTAT
        LDA TEMPB
        ADD PL
        STA TEMPB
        LDA TEMPA
        ADC PH
        STA TEMPA
ROTAT   ROR TEMPA
        ROR TEMPB
        ROR QH
        ROR QL
        DECX
        BNE MPYNXT
        RTS
*
```

```
        FCB 8                 <NUMBER>
        FCC '<NU'
        FDB WORD-6            link to WORD
NUM8    LDX SP                pop stack into GET
        LDA SP0,X
        INCX
        STA GET+1
        LDA SP0,X
        INCX
        STA GET+2
        STX SP
        CLRX                  Put char count into COUNTR
        JSR GET
        STA COUNTR
        TSTA                  count = 0 ?
        BEQ NOTNO
        INCX
        CLR TEMP              TEMP is the sign flag
        CLR QH
        CLR QL
        CLR PH                Set P = BASE
        LDA BASE
        STA PL
        JSR GET               Get first char
        CMP #$2D              = '-' ?
        BNE NUMSKP
        DEC TEMP              Minus flag = TRUE
        INCX                  bump X
        CPX COUNTR            X > COUNTR ?
        BHI NOTNO
        JSR GET
NUMSKP  INCX                  at this point X points 1 past char in A
        SUB #$30
        BMI NOTNO             if negative, not a number
        CMP #$0A              less than 10 ?
        BMI NUMB
        CMP #$11              valid char?
        BMI NOTNO
        SUB #7
NUMB    CMP BASE              valid for this base ?
        BLO ANUMB
NOTNO   CLRA                  NOPE, push a FALSE
NUMXT   LDX SP
        DECX
        STA SP0,X
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
ANUMB   STX XTEMP             save X in XTEMP
        STA ATEMP             and A in ATEMP
        JSR MPY16             Q =          Q * BASE
        LDA ATEMP             get A back
        ADD QL                Q = Q + A
        STA QL
        LDA QH
        ADC #0
        STA QH
        LDX XTEMP             get X back
        CPX COUNTR            X > COUNTR ?
        BHI NUMOK
        JSR GET
        BRA NUMSKP
NUMOK   LDA TEMP              number OK, now check sign
        TSTA
        BEQ NUMPOS
        CLRA
        NEG QL
        SBC QH
        STA QH
NUMPOS  LDX SP                push number at Q and flag
        LDA QL
        DECX
        STA SP0,X
        LDA QH
        DECX
        STA SP0,X
        LDA #$FF              TRUE flag
        BRA NUMXT+2
*
        FCB 4                 DROP
        FCC 'DRO'
        FDB NUM8-6            link to <NUMBER>
DROP    LDX SP
        INCX
        INCX
        STX SP
        JMP NEXT
*
        FCB 2                 C@
        FCC 'C@ '
        FDB DROP-6            link to DROP
CFCH    LDX SP
        LDA SP0,X
        INCX
        STA GET+1
        LDA SP0,X
        STA GET+2
        STX SP
        CLRX                  get the byte
        JSR GET
        LDX SP
```

(continued on next page)

the HD test

Professional Test/Format Program For Hard Drives in PC/XT/AT

- Setup interleave, step rate, etc.
- Surface analysis to flag bad tracks
- Load/save setup and bad track files
- Now supports AUTOCONFIG!
- Menu driven, with help windows
- Free PARK program included!
- Order HDTEST today for only $99!

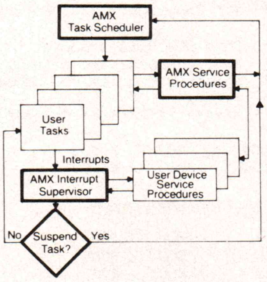
Proto PC inc.
612-644-4660
2424 Territorial Road, St. Paul, MN 55114
Telex 910-380-7623

Circle **no. 295** on reader service card.

8080 Z80 6809 AMX 8086 8088 Now 68000

Real-Time Multitasking Executive

- No royalties
- Source code included
- Fault free operation
- Ideal for process control
- Timing control provided
- Low interrupt overhead
- Inter-task messages



Options:

- Resource Manager
- Buffer Manager
- Integer Math Library

AMX is TM of KADAK Products Ltd.
CP/M-80 is TM of Digital Research Corp.
IBM, PC DOS are TM of IBM Corp.

- Language Interfaces :
  C Pascal
  PL/M Fortran
- DOS File Access :
  CP/M-80
  IBM PC DOS

AMX for	8080	$ 800 US
	8086	950
	6809	950
	68000	1600
Manual (specify processor)		75

KADAK Products Ltd.
(604) 734-2796
Telex: 04-55670
206–1847 W. Broadway, Vancouver, B.C., Canada V6J 1Y5

Circle **no. 210** on reader service card.

Listing One

(Listing continued, text begins on page 40.)

```
        STA SP0,X
        CLRA                    zero high byte
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 1                   @
        FCC '@  '
        FDB CFCH-6              link to C@
FTCH    LDX SP
        LDA SP0,X
        INCX
        STA GET+1
        LDA SP0,X
        STA GET+2
        STX SP
        LDX #1                  get low byte
        JSR GET
        LDX SP
        STA SP0,X
        CLRX                    get high byte
        JSR GET
        LDX SP
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 2                   DP
        FCC 'DP '
        FDB FTCH-6              link to @
DP2     LDX SP                  push address of DP to stack
        LDA #DP                 this routine knows that DP is on
        DECX                                          page zero
        STA SP0,X
        CLRA
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 4                   HERE
        FCC 'HER'
        FDB DP2-6               Link to DP
HERE    JMP DOCOL
        FDB DP2
        FDB FTCH
        FDB EXIT
*
        FCB 3                   NOT
        FCC 'NOT'
```

End Listing One

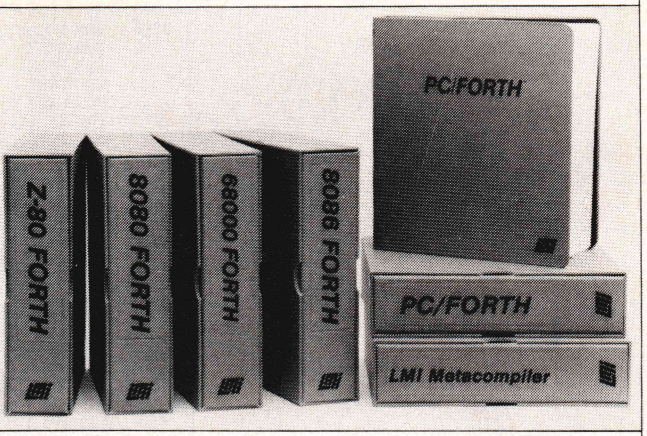
Listing Two

```
****************************************************************
*      RAFOS FORTH V1.0       26 March  1986                   *
*                                                              *
*   --- ROM #2 of 2                                            *
*                                                              *
*  (C) Copyright 1986, Everett Carter. All rights reserved.    *
*                                                              *
*  This FORTH is a subset of the FORTH-79 standard.            *
*  Some changes have been made in order to save on             *
*  space in the limited memory of the float.                   *
*                                                              *
****************************************************************
*
ROM      EQU     $1000          ROM #2 start address
ROM1     EQU     $1800          ROM #1 start address
*
         ORG     ROM
*
CR       EQU     $0D            CARRIAGE RETURN
LF       EQU     $0A            LINE FEED
BL       EQU     $20            BLANK
BS       EQU     $08            Back Space
DEL      EQU     $7F            Delete
*
DDR      EQU     4              DATA DIRECTION REGISTER OFFSET
*
PORTA    EQU     0              I/O PORT 0
PORTB    EQU     1              I/O PORT 1
PUT      EQU     PORTB          SERIAL I/O PORT
*
INITSP   EQU     $7F            INITIAL STACK POINTER VALUE
STACK    EQU     INITSP-5       TOP OF STACK
MEMSIZ   EQU     $2000          MEMORY ADDRESS SPACE SIZE
*
SP0      EQU     $0F00
RP0      EQU     $0E00
TIB      EQU     $0D80
*
*
*
*       RAM VARIABLES
*
*
ATEMP    EQU     $10            TEMP USED IN PUTDEC
XTEMP    EQU     $11            INDEX TEMPORARY
GETR     EQU     $12            PICK & DROP TEMPORARY
COUNT    EQU     $16            NUMBER OF BITS LEFT TO get/send
CHAR     EQU     $17            Current input/output character
*
BYTCNT   EQU     $1E            bytcnt.
WTIME    EQU     $20            TIMER INTERRUPT FROM WAIT STATE
*
*
PH       EQU $21                MISC SCRATCH AREAS
PL       EQU $22
TEMPA    EQU $23
TEMPB    EQU $24
QH       EQU $25
QL       EQU $26
TEMP     EQU $27
TERM     EQU $28
*
*
IN       EQU $29                   Where FORTH will look for input
OUT      EQU $2A
COUNTR   EQU $2B
DP       EQU $2C                The initial Dictionary pointer
START    EQU $2E                The start up vector
*
*
IP       EQU $30                   THE FORTH INSTRUCTION POINTER
RP       EQU $32                   THE RETURN POINTER OFFSET
SP       EQU $33                   THE STACK POINTER OFFSET
BASE     EQU $34
*
USER     EQU $35                   The space for USER variables
FENCE    EQU 0                     USER + 0
STATE    EQU 2                     USER + 2
FORTH    EQU 4                     USER + 4
CONTEXT  EQU 6                     USER + 6
CURRENT  EQU 8                     USER + 8
HLD      EQU $0A                   USER + $0A
*
*
*  List of previous FORTH words (ROM 1)
*
DOCOL   EQU $80                 DOCOL
DOCOL1  EQU DOCOL
NEXT    EQU $009E               NEXT
NEXT1   EQU $00CE
LOAD    EQU $00D1
GET     EQU $00D5
TYPE    EQU $00DF               TYPE
FIN6    EQU $0116               <FIND>
BRAN    EQU $01A9               BRAN
ZBRAN   EQU $19D2               ZBRANCH
ZBREX   EQU ZBRAN+$12
EXIT    EQU $19F8               EXIT
EXE7    EQU $1A10               EXECUTE
INLINE  EQU $1A22
EMIT    EQU $1A79               EMIT
BL2     EQU $1A8D               BL
WORD    EQU $1AA4               WORD
MPY16   EQU $1AFD
NUM8    EQU $1B27               <NUMBER>
DROP    EQU $1BBE               DROP
CFCH    EQU $1BCC               C@
FTCH    EQU $1BF2               @
DP2     EQU $1C1D               DP
HERE    EQU $1C34               HERE
NOT3    EQU $1C42               NOT
ONEP    EQU $1C5B               1+
HLD3    EQU $1C78               HLD
DOUSE   EQU $1C7A               DOUSE
STA5    EQU $1C91               STATE
CON7    EQU $1C9B               CONTEXT
CUR7    EQU $1CA5               CURRENT
FOR5    EQU $1CAF               FORTH
STO     EQU $1CB9               !
CSTO    EQU $1CE4               C!
COMA    EQU $1D04               ,
CCOMA   EQU $1D37               C,
DUP3    EQU $1D55               DUP
PLSTO   EQU $1D75               +!
LAT6    EQU $1DAE               LATEST
ALL5    EQU $1DBE               ALLOT
LIT3    EQU $1DCC               LIT
SWAP    EQU $1F07               SWAP
CRE6    EQU $1FDE               CREATE
*
*
*
LOK     EQU $1E53
OK      EQU LOK+1
*
```

```
*
OUTER EQU $1E57
*
COLD  EQU $1EA5
WARM  EQU $1EE6
GETC  EQU $1F43
GETCHAR  EQU GETC
PUTC   EQU $1F7A
OUTCHAR  EQU PUTC
WAIT   EQU $1FA8
DELAY  EQU WAIT
CRLF   EQU $1FC3
*
RESET  EQU COLD
*
***********************************************************
*
QUES   LDA DP           checks for errors at end of OUTER
       STA GET+1
       LDA DP+1
       STA GET+2
       LDX #1
       JSR GET
       CMP #$80         = buffer end ?
       BNE QERR
QEXIT  LDX SP
*
H1     EQU LOK/$100*$100
L      EQU LOK-H1
H      EQU LOK/$100
*
       LDA #L
       DECX
       STA SP0,X
       LDA #H
       DECX
       STA SP0,X
       STX SP
       LDA START
       STA IP
       LDA START+1
       STA IP+1
       JMP NEXT
QERR   LDA DP
       STA LOAD+1
       LDA DP+1
       STA LOAD+2
       CLRX
       JSR GET
       INCA
       INCA
       JSR LOAD
       TAX
       LDA #$3F         A='?'
       JSR LOAD
       LDX SP
       LDA DP+1
       DECX
       STA SP0,X
       LDA DP
       DECX
       STA SP0,X
       STX SP
       JMP WARM
*
       FCB 4            QUIT
       FCC 'QUI'
       FDB CRE6-6       link to CREATE
QUIT   BRA QEXIT
*
       FCB 6            TOGGLE
       FCC 'TOG'
       FDB QUIT-6       link to QUIT
TOG6   LDX SP
       INCX             drop high byte
       LDA SP0,X
       INCX
       STA ATEMP
       LDA SP0,X        get addr
       INCX
       STA LOAD+1
       STA GET+1
       LDA SP0,X
       INCX
       STX SP
       STA LOAD+2
       STA GET+2
       CLRX
       JSR GET
       EOR ATEMP
       JSR LOAD
       JMP NEXT
*
       FCB 9            IMMEDIATE
       FCC 'IMM'
       FDB TOG6-6       link to TOGGLE
IMM9   JMP DOCOL
       FDB LAT6
       FDB LIT3
       FDB $80
       FDB TOG6
       FDB EXIT
```

(continued on next page)

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers
- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler
- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler
- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:
- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

LMI ***Laboratory Microsystems Incorporated***
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Circle **no. 106** on reader service card.

Listing One (Listing continued, text begins on page 40.)

```
*
        FCB 5           -FIND
        FCC '-FI'
        FDB IMM9-6      link to IMMEDIATE
DFND    JMP DOCOL
        FDB BL2
        FDB WORD
        FDB CON7
        FDB FTCH
        FDB FTCH
        FDB FIN6
        FDB EXIT
*
        FCB 5           COUNT
        FCC 'COU'
        FDB DFND-6      link to -FIND
COU5    JMP DOCOL
        FDB DUP3
        FDB ONEP
        FDB SWAP
        FDB CFCH
        FDB EXIT
*
        FCB 1           0
        FCC '0 '
        FDB COU5-6      link to COUNT
ZERO    LDX SP
        CLRA
        DECX
        STA SP0,X
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 1           1
        FCC '1 '
        FDB ZERO-6      link to 0
ONE     LDX SP
        LDA #1
        DECX
        STA SP0,X
        DECX
        CLRA
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 1           2
        FCC '2 '
        FDB ONE-6       link to 1
TWO     LDX SP
        LDA #2
        DECX
        STA SP0,X
        CLRA
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 2           2+
        FCC '2+ '
        FDB TWO-6       link to 2
TWOP    LDX SP
        INCX            point to low
        LDA #2
        ADD SP0,X
        STA SP0,X
        DECX            point to high
        CLRA            A=0 note carry
        ADC SP0,X       is not affected
        STA SP0,X
        JMP NEXT
*
        FCB $81         [  (IMMEDIATE)
        FCC '[ '
        FDB TWOP-6      link to 2+
LBRAK   JMP DOCOL
        FDB ZERO
        FDB STA5
        FDB STO
        FDB EXIT
*
        FCB 1           ]
        FCC '] '
        FDB LBRAK-6     link to [
RBRAK   JMP DOCOL
        FDB LIT3
        FDB $C0
        FDB STA5
        FDB STO
        FDB EXIT
*
        FCB 11          DEFINITIONS
        FCC 'DEF'
        FDB RBRAK-6     link to ]
DEFS    JMP DOCOL
        FDB CON7
        FDB FTCH
        FDB CUR7
        FDB STO
        FDB EXIT
*
        FCB 1           +
        FCC '+ '
        FDB DEFS-6      link to DEFINITIONS
PLUS    LDX SP
        LDA SP0,X
        INCX
        STA PH
        LDA SP0,X
        INCX
        STX SP
        INCX            point to low on stack
        ADD SP0,X
        STA SP0,X
        LDX SP
        LDA PH
        ADC SP0,X
        STA SP0,X
        JMP NEXT
*
        FCB 1           -
        FCC '- '
        FDB PLUS-6      link to +
MINUS   LDX SP
        LDA SP0,X
        INCX
        STA PH
        LDA SP0,X
        INCX
        STA PL
        STX SP
        INCX            point to low on stack
        LDA SP0,X
        SUB PL
        STA SP0,X
        LDX SP
        LDA SP0,X
        SBC PH
        STA SP0,X
        JMP NEXT
*
        FCB 2           U*
        FCC 'U* '
        FDB MINUS-6     link to -
UMULT   LDX SP
        LDA SP0,X
        INCX
        STA PH
        LDA SP0,X
        INCX
        STA PL
        LDA SP0,X
        INCX
        STA QH
        LDA SP0,X
        STA QL
        STX XTEMP
        JSR MPY16
        LDX XTEMP
        LDA QL               push low word
        STA SP0,X
        LDA QH
        DECX
        STA SP0,X
        LDA TEMPB            push high word
        DECX
        STA SP0,X
        LDA TEMPA
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
SEC     EQU PH
*
DIV16   LDA SEC+2            Dividend: (H to L)
        LDX SEC              PL,PH,TEMPB,TEMPA
        STX SEC+2            (SEC +1,+0,+3,+2)
        LSLA
        STA SEC
        LDA SEC+3            Divisor is QH,QL
        LDX SEC+1
        STX SEC+3
        ROLA
        STA SEC+1
        LDA #$10
        STA TEMP
DBEG    ROL SEC+2
        ROL SEC+3
        LDA SEC+2
        SUB QL
        TAX
        LDA SEC+3
        SBC QH
        BCS DSKIP
        STX SEC+2
        STA SEC+3
        SEC
        BRA DSKIP+1
DSKIP   CLC
        ROL SEC         Quotient (H,L) is
        ROL SEC+1          SEC+1,SEC
        DEC TEMP        Remainder (H,L) is
        BNE DBEG           SEC+3,SEC+2
        RTS
*
        FCB 5           U/MOD
        FCC 'U/M'
        FDB UMULT-6     link to U*
UDMD    LDX SP
        LDA SP0,X
        INCX
        STA QH          divisor to Q
        LDA SP0,X
        INCX
        STA QL
        LDA SP0,X
        INCX
        STA SEC+1         high word to P
        LDA SP0,X         (SEC +1, +0)
        INCX
        STA SEC
        LDA SP0,X
        INCX
        STA SEC+3         low word to TEMPA/B
        LDA SP0,X         (SEC +3, +2)
        STA SEC+2
        STX SP
        BSR DIV16
        LDX SP
        LDA SEC+2         push remainder
        STA SP0,X         (HL: SEC+3, +2)
        LDA SEC+3
        DECX
        STA SP0,X
        LDA SEC           push quotient
        DECX              (HL: SEC+1, +0)
        STA SP0,X
        LDA SEC+1
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 4           S->D
        FCC 'S->'
        FDB UDMD-6      link to U/MOD
STOD    LDX SP
        LDA SP0,X
        TSTA
        BPL SPOS
        LDA #$FF
        BRA SEXIT
SPOS    CLRA
SEXIT   DECX
        STA SP0,X
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 3           PAD
        FCC 'PAD'
        FDB STOD-6      link to S->D
PAD     JMP DOCOL
        FDB HERE
        FDB LIT3
        FDB $0044
        FDB PLUS
        FDB EXIT
*
        FCB 2           <#
        FCC '<# '
        FDB PAD-6       link to PAD
LSHP    JMP DOCOL
        FDB PAD
        FDB HLD3
        FDB STO
        FDB EXIT
*
        FCB 4           OVER
        FCC 'OVE'
        FDB LSHP-6      link to <#
OVER    LDX SP
        INCX
        INCX
        LDA SP0,X
        INCX
```

```
        STA ATEMP
        LDA SP0,X
        LDX SP
        DECX
        STA SP0,X
        LDA ATEMP
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 2           #>
        FCC '#> '
        FDB OVER-6      link to OVER
SPGR    JMP DOCOL
        FDB DROP
        FDB DROP
        FDB HLD3
        FDB FTCH
        FDB PAD
        FDB OVER
        FDB MINUS
        FDB EXIT
*
        FCB 2           >R
        FCC '>R '
        FDB SPGR-6      link to #>
TOR     LDX SP
        LDA SP0,X
        INCX
        STA ATEMP
        LDA SP0,X
        INCX
        STX SP
        LDX RP
        DECX
        STA RP0,X
        LDA ATEMP
        DECX
        STA RP0,X
        STX RP
        JMP NEXT
*
        FCB 2           R>
        FCC 'R> '
        FDB TOR-6       link to >R
RTO     LDX RP
        LDA RP0,X
        INCX
        STA ATEMP
        LDA RP0,X
        INCX
        STX RP
        LDX SP
        DECX
        STA SP0,X
        LDA ATEMP
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 2           R@
        FCC 'R@ '
        FDB RTO-6       link to R>
RFTCH   LDX RP
        LDA RP0,X       pop high
        INCX
        STA ATEMP
        LDA RP0,X       pop low
        LDX SP
        DECX
        STA SP0,X       push low
        LDA ATEMP
        DECX
        STA SP0,X       push high
        STX SP
        JMP NEXT
*
        FCB 3           ROT
        FCC 'ROT'
        FDB RFTCH-6     link to R@
ROT     JMP DOCOL
        FDB TOR
        FDB SWAP
        FDB RTO
        FDB SWAP
        FDB EXIT
*
        FCB 4           HOLD
        FCC 'HOL'
        FDB ROT-6       link to ROT
HOLD    JMP DOCOL
        FDB LIT3
        FDB $FFFF        ( -1 )
        FDB HLD3
        FDB PLSTO
        FDB HLD3
        FDB FTCH
        FDB CSTO
        FDB EXIT
*
        FCB 5           M/MOD
        FCC 'M/M'
        FDB HOLD-6      link to HOLD
MDM5    JMP DOCOL
        FDB TOR
        FDB ZERO
        FDB RFTCH
        FDB UDMD
        FDB RTO
        FDB SWAP
        FDB TOR
        FDB UDMD
        FDB RTO
        FDB EXIT
*
        FCB 4           BASE
        FCC 'BAS'
        FDB MDM5-6      link to M/MOD
BAS4    JMP DOCOL
        FDB LIT3
        FDB BASE
        FDB EXIT
*
        FCB 6           SMUDGE
        FCC 'SMU'
        FDB BAS4-6      link to BASE
SMUDG   JMP DOCOL
        FDB LAT6
        FDB LIT3
        FDB $0020
        FDB TOG6
        FDB EXIT
*
        FCB 3           ABS
        FCC 'ABS'
        FDB SMUDG-6     link to SMUDGE
ABS     LDX SP
        LDA SP0,X
        TSTA
        BPL ABXIT
        COMA
        STA SP0,X
        INCX
        LDA SP0,X
        NEGA
        STA SP0,X
ABXIT   JMP NEXT
*
        FCB 2           0<
        FCC '0< '
        FDB ABS-6       link to ABS
ZLESS   LDX SP
        LDA SP0,X
        TSTA
        BPL ZLPOS
        LDA #$FF
        BRA ZLXIT
ZLPOS   CLRA
ZLXIT   STA SP0,X
        INCX
        STA SP0,X
        JMP NEXT
*
        FCB 2           0=
        FCC '0= '
        FDB ZLESS-6     link to 0<
ZEQ     LDX SP
        LDA SP0,X
        INCX
        ORA SP0,X
        BNE ZEN
        LDA #$FF
        BRA ZEXIT
ZEN     CLRA
ZEXIT   STA SP0,X
        DECX
        STA SP0,X
        JMP NEXT
*
        FCB 1           <
        FCC '<  '
        FDB ZEQ-6       link to 0=
LESS    JMP DOCOL
        FDB MINUS
        FDB ZLESS
        FDB EXIT
*
        FCB 1           >
        FCC '>  '
        FDB LESS-6      link to <
GREAT   JMP DOCOL
        FDB SWAP
        FDB LESS
        FDB EXIT
*
        FCB 1           =
        FCC '=  '
        FDB GREAT-6     link to >
EQUAL   JMP DOCOL
        FDB MINUS
        FDB ZEQ
        FDB EXIT
```

(continued on next page)

. . . *Software Developers* . . .

Develop your software to work with most of the desktop printers on the market?

Now, CARDINAL POINT INCORPORATED has published Volume II of the ***PROGRAMMERS' HANDBOOK OF COMPUTER PRINTER COMMANDS II, for models as new as 1985.*** Volume II compliments (but does not replace) book one and adds more detailed ESCAPE and Control Codes sequences for more printers by the leading printer manufacturers. Each book is 5½" x 8½", spiral bond, and contains an easy to use table-format. Codes are listed in text form with hex and decimal equivalents and detailed function description.

These two volumes should be a part of your reference library.

1. **Volume I (models through 1984)** **$37.95**
2. **Volume II (models as new as 1985)** **$26.95**
3. **2 book set (Vols. I & II)** **$58.95**

(Indiana residents add 5% sales tax.
Please include $2.50 shpg/hdlg)

TO ORDER CALL:

1-800-628-2828 ext. 534 (24 Hrs)

Quantity Orders or Dealers may call the number below collect. For detailed information, call or write:

Cardinal Point INCORPORATED
P.O. Box 596, Ellettsville, IN 47429
(812) 876-7811 (M-F, 9-5 EST)
We accept MC, VISA, MO-same day shpg.
COD — $2 extra. CK's — Allow extra 14 days.

Circle **no. 246** on reader service card.

IBM / PC CROSS ASSEMBLERS

Assemblers now available include:

Chip	*Chip*	*Chip*
1802/1805	*HD64180*	*8051*
9900/9995	*NSC800*	*6804*
6500/01/02	*F8, 3870*	*6805*
6800/01/02	*Z8*	*6809*
6800/08/10	*Z80*	*6811*
8048/49/50/42	*Z8000*	*8085*
65C02/C102/C112	*6803/08*	*6301*

RELATIONAL MEMORY SYSTEMS, INC.
P. O. Box 6719
San Jose, California 95150

Tel: (408) 265-5411
CPM80, MPM, ISIS Versions Available

relms

Circle **no. 120** on reader service card.

Incredible SwyftWare!™

It is not possible in an ad to really explain this Apple II (e or c) product which outperforms all other software in speed and simplicity. It does no-frills word processing, information retrieval, telecommunications and many other tasks. Reviewers call it "foolproof and frustration free" and "blindingly fast." It appeals to beginners and experts. Jef Raskin, who invented the Macintosh and who later created SwyftWare, has written up the hows and whys of this product. Please write for a copy. No charge. Or order SwyftWare and see for yourself. At $89.95 and with a 30-day refund policy, there's no risk.

US 800/982-5600 CA 800/562-7400

Information Appliance
1014 Hamilton Ct., Menlo Park, CA 94025

Listing Two (Listing continued, text begins on page 40.)

```
*
        FCB 4             SIGN
        FCC 'SIG'
        FDB EQUAL-6       link to =
SIGN    JMP DOCOL
        FDB ZLESS
        FDB ZBRAN
        FDB $0008
        FDB LIT3
        FDB $002D
        FDB HOLD
        FDB EXIT
*
        FCB 6             NEGATE
        FCC 'NEG'
        FDB SIGN-6        link to SIGN
NEG6    LDX SP
        LDA SP0,X
        COMA
        STA SP0,X
        INCX
        LDA SP0,X
        NEGA
        STA SP0,X
        JMP NEXT
*
        FCB 2             +-
        FCC '+- '
        FDB NEG6-6        link to NEGATE
PIMI    JMP DOCOL
        FDB ZLESS
        FDB ZBRAN
        FDB $0004
        FDB NEG6
        FDB EXIT
*
        FCB 1             #
        FCC '#  '
        FDB PIMI-6        link to +-
SHARP   JMP DOCOL
        FDB BAS4
        FDB CFCH
        FDB MDM5
        FDB ROT
        FDB LIT3
        FDB $0009
        FDB OVER
        FDB LESS
        FDB ZBRAN
        FDB $0008
        FDB LIT3
        FDB $0007
        FDB PLUS
        FDB LIT3
        FDB $0030
        FDB PLUS
        FDB HOLD
        FDB EXIT
*
        FCB 2             OR
        FCC 'OR '
        FDB SHARP-6       link to #
OR2     LDX SP
        LDA SP0,X
        INCX
        INCX
        ORA SP0,X
        STA SP0,X
        DECX
        LDA SP0,X
        INCX
        INCX
        ORA SP0,X
        STA SP0,X
        DECX
        STX SP
        JMP NEXT
*
        FCB 3             AND
        FCC 'AND'
        FDB OR2-6         link to OR
AND3    LDX SP
        LDA SP0,X
        INCX
        INCX
        AND SP0,X
        STA SP0,X
        DECX
        LDA SP0,X
        INCX
        INCX
        AND SP0,X
        STA SP0,X
        DECX
        STX SP
        JMP NEXT
*
        FCB 3             XOR
        FCC 'XOR'
        FDB AND3-6        link to AND
XOR3    LDX SP
        LDA SP0,X
        INCX
        INCX
        EOR SP0,X
        STA SP0,X
        DECX
        LDA SP0,X
        INCX
        INCX
        EOR SP0,X
        STA SP0,X
        DECX
        STX SP
        JMP NEXT
*
        FCB 4             DDUP
        FCC 'DDU'
        FDB XOR3-6        link to XOR
DDUP    JMP DOCOL
        FDB OVER
        FDB OVER
        FDB EXIT
*
        FCB 2             #S
        FCC '#S '
        FDB DDUP-6        link to DDUP
SHRPS   JMP DOCOL
        FDB SHARP
        FDB DDUP
        FDB OR2
        FDB ZEQ
        FDB ZBRAN
        FDB $FFF6
        FDB EXIT
*
        FCB 1             .
        FCC '.  '
        FDB SHRPS-6       link to #S
DOT     JMP DOCOL
        FDB DUP3
        FDB DUP3
        FDB ABS
        FDB STOD
        FDB LSHP
        FDB SHRPS
        FDB ROT
        FDB SIGN
        FDB SPGR
        FDB TYPE
        FDB DROP
        FDB BL2
        FDB EMIT
        FDB EXIT
*
        FCB 7             COMPILE
        FCC 'COM'
        FDB DOT-6         link to .
COMP    JMP DOCOL
        FDB RTO
        FDB DUP3
        FDB TWOP
        FDB TOR
        FDB FTCH
        FDB COMA
        FDB EXIT
*
        FCB $81           ;  (IMMEDIATE)
        FCC ';  '
        FDB COMP-6        link to COMPILE
SEMI    JMP DOCOL
        FDB COMP
        FDB EXIT
        FDB SMUDG
        FDB LBRAK
        FDB EXIT
*
        FCB 1             :
        FCC ':  '
        FDB SEMI-6        link to ;
COLON   JMP DOCOL
        FDB CUR7
        FDB FTCH
        FDB CON7
        FDB STO
        FDB CRE6
        FDB SMUDG
        FDB COMP
        JMP DOCOL
        FDB RBRAK
        FDB EXIT
*
        FCB $81           '  (IMMEDIATE)
        FCB $27
        FCC '  '
        FDB COLON-6       link to :
TICK    JMP DOCOL
        FDB DFND
        FDB ZBRAN
        FDB $0006
        FDB DROP
        FDB EXIT
        FDB QUES
*
DOVAR   LDX SP
        LDA NEXT1+2       low byte of W
        ADD #3
        DECX
        STA SP0,X
        LDA NEXT1+1       high byte of W
        ADC #0
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 8             VARIABLE
        FCC 'VAR'
        FDB TICK-6        link to '
VAR8    JMP DOCOL
        FDB CRE6
        FDB LIT3          compile a jump
        FDB $00CC         to DOVAR
        FDB CCOMA
        FDB COMP
        FDB DOVAR
        FDB TWO
        FDB ALL5
        FDB EXIT
*
DOCON   LDA NEXT1+2       put W+2 into GET
        ADD #3
        STA GET+2
        LDA NEXT1+1
        ADC #0
        STA GET+1
        LDX #1
        JSR GET           get low byte of
                                constant
        LDX SP
        DECX
        STA SP0,X
        STX SP
        CLRX
        JSR GET           then the high byte
        LDX SP
        DECX
        STA SP0,X
        STX SP
        JMP NEXT
*
        FCB 8             CONSTANT
        FCC 'CON'
        FDB VAR8-6        link to VARIABLE
CON8    JMP DOCOL
        FDB CRE6
        FDB LIT3          compile a jump
                               to DOCON
        FDB $00CC
        FDB CCOMA
        FDB COMP
        FDB DOCON
        FDB COMA
        FDB EXIT
*
        FCB 1             *
        FCC '*  '
        FDB CON8-6        link to CONSTANT
MULT    JMP DOCOL
        FDB UMULT
        FDB DROP
        FDB EXIT
*
        FCB $89           [COMPILE]  IMMEDIATE
        FCC '[CO'
        FDB MULT-6        link to *
BCOM9   JMP DOCOL
        FDB TICK
        FDB COMA
        FDB EXIT
*
        FCB $85           BEGIN     IMMEDIATE
        FCC 'BEG'
        FDB BCOM9-6       link to [COMPILE]
BEGN    JMP DOCOL
        FDB HERE
        FDB EXIT
*
        FCB $85           AGAIN     IMMEDIATE
        FCC 'AGA'
        FDB BEGN-6        link to BEGIN
```

```
AGAIN  JMP DOCOL
       FDB COMP
       FDB BRAN
       FDB HERE
       FDB MINUS
       FDB COMA
       FDB EXIT
*
       FCB $85            UNTIL      IMMEDIATE
       FCC 'UNT'
       FDB AGAIN-6        link to AGAIN
UNTIL  JMP DOCOL
       FDB COMP
       FDB ZBRAN
       FDB HERE
       FDB MINUS
       FDB COMA
       FDB EXIT
*
       FCB $82            IF         IMMEDIATE
       FCC 'IF '
       FDB UNTIL-6        link to UNTIL
IF2    JMP DOCOL
       FDB COMP
       FDB ZBRAN
       FDB HERE
       FDB ZERO
       FDB COMA
       FDB EXIT
*
       FCB $84            THEN       IMMEDIATE
       FCC 'THE'
       FDB IF2-6          link to IF
THEN   JMP DOCOL
       FDB HERE
       FDB OVER
       FDB MINUS
       FDB SWAP
       FDB STO
       FDB EXIT
*
       FCB $84            ELSE       IMMEDIATE
       FCC 'ELS'
       FDB THEN-6         link to THEN
ELSE   JMP DOCOL
       FDB COMP
       FDB BRAN
       FDB HERE
       FDB ZERO
       FDB COMA
       FDB SWAP
       FDB THEN
       FDB EXIT
*
       FCB $85            WHILE      IMMEDIATE
       FCC 'WHI'
       FDB ELSE-6         link to ELSE
WHILE  JMP DOCOL
       FDB IF2
       FDB EXIT
*
       FCB $86            REPEAT     IMMEDIATE
       FCC 'REP'
       FDB WHILE-6        link to WHILE
REPET  JMP DOCOL
       FDB TOR
       FDB AGAIN
       FDB RTO
       FDB THEN
       FDB EXIT
*
       FCB 4              <.">
       FCC '<."'
       FDB REPET-6        link to REPEAT
BDOTQ  JMP DOCOL
       FDB RFTCH
       FDB COU5
       FDB DUP3
       FDB ONEP
       FDB RTO
       FDB PLUS
       FDB TOR
       FDB TYPE
       FDB EXIT
*
       FCB 3              TIB
       FCC 'TIB'
       FDB BDOTQ-6        link to <.">
TIB3   JMP DOCOL
       FDB LIT3
       FDB TIB
       FDB EXIT
*
       FCB 3              >IN
       FCC '>IN'
       FDB TIB3-6         link to TIB
FRIN   JMP DOCOL
       FDB LIT3
       FDB IN
       FDB EXIT
*
       FCB 7              'STREAM
       FCB $27
```

(continued on next page)

The Best Debuggers. Period.

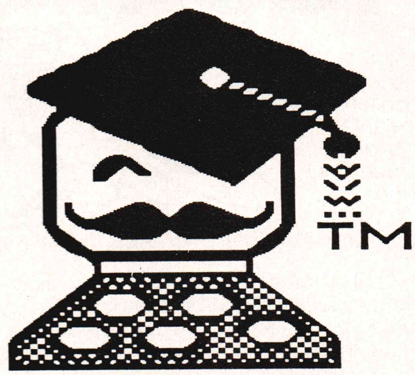
DSD86, The PC-DOS Debugger *69.95*
DSD87, The PC-DOS Debugger with 8087 Support . *99.95*
DSD80, The CP/M Debugger *125.00*

SoftAdvances

P.O. Box 49473 • Austin, Texas 78765 • (512) 478-4763
1-800-232-8088

Circle **no. 83** on reader service card.

MacTutor
The Macintosh Programming Journal



Simply the finest monthly technical Journal available for the Macintosh. Contains no fluff, only programs in C, Asm, Pascal, Basic & Forth. US 3rd class: $30; US 1st class, or Canada: $45. All Overseas: $60. (714) 630-3730.

Subscribe Today!

MacTutor
P.O. Box 400
Placentia, CA. 92670

Circle **no. 187** on reader service card.

Now! Automatic time and datestamping for CP/M® 2.2

DATESTAMPER™

- Extends CP/M 2.2 to automatically record date and time a file is created, read and modified. DateStamper reads the exact time from a real-time clock if you have one, or records the order in which you use files each day.
- Datestamping information is contained in a separate file and read by our directory utility, SDD.COM. Powerful DATSWEEP file management utility also included.
- Simple menu-driven installation. Large library of clocks supplied, with provision to add others. Many options configurable to your individual requirements.
- Disks prepared for datestamping are fully compatible with standard CP/M. DateStamper also runs with all versions of ZCPR, Z-RDOS and many other CP/M-80 modifications.

CP/M is a registered trademark of Digital Research, Inc.

Avoid erasing the wrong files!

Back up files by time and date!

"DateStamper... is a real winner."
Bruce Morgen, Users Guide, Jul-Aug. 1985

Plu*Perfect Systems

8" SSSD, Kaypro, Osborne, H/Z-89 formats $49
(Most other formats, add $5)
Shipping & handling $3
California residents add 6% sales tax

MasterCard & Visa accepted

Write or call for further information
(714) 659-4432

BOX 1494, IDYLLWILD, CA 92349

Circle no. 193 on reader service card.

TOOLS FOR TODAY

The MasterFORTH programming system gets your application running in the least amount of time. How? With a built-in user interface. With an interactive debugger. With a resident assembler–linkage is instantaneous.

MasterFORTH's target application generation system (TAGS) can produce a ROM-able image in seconds. Its unique symbol table identifies seldom-used code so you can pack the most power in the least amount of memory.

Choose your development system. We support the IBM PC family, Apple's Macintosh, CP/M systems and others. Need to program a controller from your IBM PC? No problem. We have cross-development sources for most popular microprocessors. You can even generate a MasterFORTH system for your target application, add a terminal and run it there. Forth is widely used in industry and aerospace for robotics and machine control.

Call or write us for details. MICROMOTION

8726 S. Sepulveda Bl., #A171
Los Angeles, CA 90045
(213) 821-4340

Circle no. 209 on reader service card.

Listing Two

(Listing continued, text begins on page 40.)

```
        FCC 'ST'
        FDB FRIN-6                  link to >IN
TSTRM   JMP DOCOL
        FDB TIB3
        FDB FRIN
        FDB CFCH
        FDB PLUS
        FDB EXIT
*
        FCB 4                        <DO>
        FCC '<DO'
        FDB TSTRM-6                  link to 'STREAM
BDO     LDA #4
        STA COUNTR
        ADD SP                       make 2 artificial pops
        STA SP
DOAGIN  LDX SP                       move limit
        DECX                         then index
        LDA SP0,X                    from SP
        STX SP                       to RP
        LDX RP
        DECX
        STA RP0,X
        STX RP
        DEC COUNTR
        BNE DOAGIN
        LDA #4                       adjust SP
        ADD SP
        STA SP
        JMP NEXT
*
        FCB 6                        <LOOP>
        FCC '<LO'
        FDB BDO-6                    link to <DO>
BLOP    CLR PH                     set increment to 1
        LDA #1
        STA PL
LOOPS   LDX RP                     increment index
        INCX                       by value
        LDA PL                     in P H/L
        ADD RP0,X
        STA RP0,X
        DECX
        LDA PH
        ADC RP0,X
        STA RP0,X
        INCX
        LDA RP0,X                test index-limit
        INCX
        INCX
        SUB RP0,X
        LDX RP
        LDA RP0,X
        INCX
        INCX
        SBC RP0,X
        EOR PH             also check increment sign
        BMI LAGIN          loop again if negative
        INCX
        INCX
        STX RP
        JMP ZBREX
LAGIN   JMP BRAN
*
        FCB 7                  <+LOOP>
        FCC '<+L'
        FDB BLOP-6             link to <LOOP>
BPLOP   LDX SP
        LDA SP0,X
        INCX
        STA PH                set increment
        LDA SP0,X             from the stack
        INCX
        STA PL
        STX SP
        BRA LOOPS
*
        FCB $82               DO           IMMEDIATE
        FCC 'DO '
        FDB BPLOP-6            link to <+LOOP>
DO      JMP DOCOL
        FDB COMP
        FDB BDO
        FDB HERE
        FDB EXIT
*
        FCB $84               LOOP          IMMEDIATE
        FCC 'LOO'
        FDB DO-6              link to DO
LOOP    JMP DOCOL
        FDB COMP
        FDB BLOP
        FDB HERE
        FDB MINUS
```

```
        FDB COMA
        FDB EXIT
*
        FCB $85              +LOOP        IMMEDIATE
        FCC '+LO'
        FDB LOOP-6           link to LOOP
PLOOP   JMP DOCOL
        FDB COMP
        FDB BPLOP
        FDB HERE
        FDB MINUS
        FDB COMA
        FDB EXIT
*
        FCB 7                DNEGATE
        FCC 'DNE'
        FDB PLOOP-6          link to +LOOP
DNEG7   LDA #3
        STA COUNTR
        LDX SP
DNLP    LDA SP0,X            ones complement
        COMA                 three bytes
        STA SP0,X
        INCX
        DEC COUNTR
        BNE DNLP
        LDA SP0,X            twos complement
        NEGA                 the fourth
        STA SP0,X
        JMP NEXT
*
        FCB $81              I   IMMEDIATE
        FCC 'I  '
        FDB DNEG7-6          link to DNEGATE
I1      JMP DOCOL
        FDB COMP
        FDB RFTCH
        FDB EXIT
*
*
*
*
           END

$
```

End Listings

WORLD'S FIRST ASSEMBLER INTERPRETER

Advanced Trace86™ 2.0

- Create .COM programs with BASIC-like commands: LOAD, SAVE, EDIT, LIST, RUN. Reloadable into AT86, complete with all labels, comments & variable names.
- Macro Assembler support-scan .MAP and .LST files for labels and variable names
- Full-screen symbolic tracing of any program.
- Powerful conditional breakpoint facilities, including hardware "button" support (if installed)
- "Screen save" option, or use dual monitors
- Hex/decimal calculator & converter
- Best 8087/80286/80287 support on the market!
- And more . . . Priced at $175.00

To order or request more information contact:

Morgan Computing Co., Inc.
(214) 245-4763
P.O. Box 112730, Carrollton, TX 75011

Circle **no. 128** on reader service card.

SEPTEMBER '86 ISSUE THEME:

ALGORITHMS

Algorithms are the building blocks of a program—the logical steps taken to solve a problem, independent of the detailed syntax of a programming language.

Make *Dr. Dobb's Journal of Software Tools* your algorithm for marketing success. *Dr. Dobb's* offers a direct line of communication with the core of the computer marketplace: the professional programmer.

Reach your target audience: call to reserve ad space now!!

ABC Audited Paid Circulation

Robert Horton
Advertising Director
415-366-3600

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

501 Galveston Drive, Redwood City, CA 94063

Up To Your Ears In Alligators?

If that sounds familiar, you need Write-Hand-Man™, the multi-function pop-up desktop organizer that works neatly with existing software for CP/M™ 2.2 and 3.0 systems. Write-Hand-Man eliminates that swamped feeling with tools that will get you organized. Write-Hand-Man comes with a 4-function, floating-point, 14 digit Calculator – Notepad – Two-week Appointment Book, File and Directory viewing – Phonebook with dialing – Cut and Paste – Key Redefinition – ASC II table. Even add your own applications.

Clear the swamp from your desktop.
Order Write-Hand-Man today. $49.95

CA residents add 6.5% tax. Sorry, no credit cards or purchase orders.
Specify: 8" or which 5" format
CP/M 2.2 or 3.0 format
30 day guarantee
™Write-Hand-Man – Poor Person Software
™CP/M – Digital Research

Poor Person
Software

Dept. 203
3721 Starr King Circle
Palo Alto, CA 94306
(415) 493-3735

Circle **no. 169** on reader service card.

Listing One (Text begins on page 46.)

```
file: WINDOW.BLK        Block: 0

                                                  cl 11/10/85
window program
by
Craig A. Lindley
Manitou Springs
Colorado
November 1985

file: WINDOW.BLK        Block: 1

\ window routines                                 cl 11/10/85
\ window load screen

warning off

dark
.( Compiling window package and demo program )
cr

2 32 thru

warning on
```

```
file: WINDOW.BLK        Block: 2

\ case statement                                  cl 11/10/85
\ Dr. Charles Eakers Forth Dimensions Vol 2, Num 3
: ?comp    state @ not abort" Compilation only" ;
: ?pairs   <>          abort" Bad CASE statement"  ;

: case     ?comp csp @ !csp 4 ; immediate
: of       4 ?pairs
           compile over compile = compile ?branch
           here 0 , compile drop 5 ; immediate
: endof    5 ?pairs compile branch here 0 ,
           swap   >resolve 4 ; immediate
: endcase 4 ?pairs compile drop
           begin sp@ csp @ <>
           while >resolve repeat
           csp ! ; immediate
```

```
file: WINDOW.BLK        Block: 3

\ window routines                                 cl 11/10/85
\ write count # of chars with attrib at cursor position
code chra                        \ char/attrib count --
 cx pop ax pop ah bl mov         \ get count in cx, attrib in bl
 bh bh xor 9 # ah mov            \ char in al, func. code in ah
 si push 16 int si pop           \ do video interrupt
 next
end-code
\ write 1 char with attrib at cursor - update cursor position
code chra+                       \ char/attrib --
 ax pop ah bl mov bh bh xor      \ char in al, attrib in bl
 1 # cx mov 9 # ah mov           \ char in al, func. code in ah
 si push 16 int                  \ count=1, write char/attrib
 3 # ah mov 16 int dl inc 2 # ah mov 16 int
 si pop next                     \ inc cursor position
end-code
```

```
file: WINDOW.BLK        Block: 4

\ window routines                                 cl 11/10/85
\ read char and attrib at cursor position
code rdchra                      \ -- char/attrib
 0 # bh mov 8 # ah mov           \ pg =0  func. code = 8
 si push 16 int si pop           \ do video interrupt
 lpush                           \ char/attrib to stk
end-code

\ put char with attrib at x,y
: putch                          \ x y char/attrib --
  >r at r> 1 chra ;

\ get char with attrib at x,y
: getch                          \ x y -- char/attrib
  at rdchra ;
```

```
file: WINDOW.BLK        Block: 5

\ window routines                                 cl 11/10/85

\ draw count # of chars/attrib starting at x,y
: draw_row                       \ x y char/attrib count --
  >r >r at r> r> chra ;

\ scroll specified window up n lines
code scrlup                      \ xul yul xlr ylr cnt attrib --
 bx pop bl bh mov di pop         \ bh attrib si # of lines
 dx pop dl dh mov ax pop al dl mov \ dx has lr x y
 cx pop cl ch mov ax pop al cl mov \ cx has ul x y
 di ax mov si push bp push       \ save regs
 6 # ah mov 16 int               \ ax # of lines func. code ah
 bp pop si pop next              \ restore forth's regs
end-code
```

```
file: WINDOW.BLK        Block: 6

\ window routines                                 cl 11/10/85
\ memory management support
\ tell DOS to allocate memory bytes
code calloc                      \ # bytes -- seg T
 bx pop 4 # cl mov bx cl shr     \          -- maxp error code F
 bx inc 72 # ah mov 33 int       \ int 21h func. code 48h
 u< if   bx push ax push ax ax xor \ if C then error
    else ax push -1 # ax mov then lpush
end-code
\ tell DOS to free memory segment
code free                        \ seg -- T
```

(continued on page 98)

WIZARD C

The new **Wizard C version 3.0** sets new records for all out speed! It leaves other C compilers in the dust! When your project depends on that last ounce of speed, choose Wizard.

The following SIEVE benchmark was run without register variable declarations on an IBM/PC with 640K memory and an 8087.

	Exec Time	Code Size	EXE Size
Wizard C 3.0	**: 6.8**	**130**	**7,766**
Microsoft	:11.5	186	7,018
Lattice	:11.8	164	20,068

Fast executable code, with multiple levels of optimization.

Six memory Models, supporting up to 1 megabytye of code and data, plus mixed model programming.

Effective error diagnosis, including multiple source file cross-checking of function calls.

A comprehensive runtime library, including fully portable C functions, MSDOS and 8086 functions.

8086, 8087, 80186 and 80286 hardware support.

Full Library Source Code included with package.

ROM based application support, including a ROM development package available to create Intel Hex files.

Fully ANSI compatible language features.

"...The compiler's performance makes it very useful in serious software development."

PC Tech Journal
January, 1986

" Wizard's got the highest marks for support."

"The Wizard Compiler had excellent diagnostics; it would be easier writing portable code with it than with any other compiler we tested."

Dr. Dobb's Journal
August, 1985

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

For MSDOS applications, we provide the most features of any C compiler, plus a full range of third party software:

- PANEL
- Greenleaf Libraries
- Essential Software Library
- PLINK86
- Pfix86plus
- Microsoft Assembler 3.0

For stand-alone applications, we supply a ROM development package that carries your program all the way to Intel Hex files ready for a PROM burner.

For debugging, the compiler emits full Intel debugging information including local symbol and type information.

Wizard C	**$450.00**
ROM Development Package	**$350.00**
Combined Package	**$750.00**

(617) 641-2379

WIZARD
SYSTEMS SOFTWARE, INC.
11 Willow Court, Arlington, MA 02174

Listing One (Listing continued, text begins on page 46.)

```
 ax pop ax es mov              \      -- error code F
 73 # ah mov 33 int            \ int 21h func. code 49h
 u< if   ax push ax ax xor     \ if C then error
    else -1 # ax mov then 1push
end-code
```

```
file: WINDOW.BLK        Block: 7

\ window routines                                    cl 11/10/85
\ memory management support
\ tell DOS to shrink or expand allociated memory segment

code setblock                  \ # bytes -- T
 cs ax mov ax es mov           \           -- maxp error code F
 bx pop 4 # cl mov bx cl shr   \ bx has # of paragraphs
 bx inc 74 # ah mov 33 int     \ int 21h func. code 4Ah
 u< if   bx push ax push ax ax xor \ if C then error
    else -1 # ax mov
    then 1push
end-code
```

```
file: WINDOW.BLK        Block: 8

\ window routines                                    cl 11/10/85
\ extended word fetch and store words
\ fetch word from extended memory
code e@                        \ seg addr -- n
 bx pop es pop                 \ seg in es addr in bx
 es: 0 [bx] ax mov             \ get the data on stk
 1push
end-code

\ store word in extended memory
code e!                        \ n seg addr --
 bx pop es pop ax pop
 ax es: 0 [bx] mov             \ store the data
 next
end-code
```

```
file: WINDOW.BLK        Block: 9

\ window routines                                    cl 11/10/85
\ read current cursor location

code rdcur                     \ -- x y
 si push 0 # bh mov 3 # ah mov \ int 10h func. code 3
 16 int  si pop ah ah xor
 dl al mov ax push dh al mov
 1push
end-code
```

```
file: WINDOW.BLK        Block: 10

\ window routines                                    cl 11/10/85
\ window control block (wcb) record layout
 0 constant ulx      2 constant uly           \ upper left corner
 4 constant width    6 constant height        \ width and height
 8 constant curx    10 constant cury          \ current cursor pos
12 constant oldx    14 constant oldy          \ old cursor pos.
16 constant bufseg  18 constant oldwcbseg     \ seg storage
20 constant attrib                            \ window attrib.

22 constant record_size                       \ size of record
15 constant boarder                           \ boarder attribute
hex
b800 constant v_seg                           \ video memory start
   variable wcbseg                            \ current wcb seg
decimal                                       \ storage
```

```
file: WINDOW.BLK        Block: 11

\ window routines                                    cl 11/10/85
\ extended memory fetch and store words

\ store word n at addr in current wcb
: wcbseg!                       \ n addr --
  wcbseg @ swap e! ;            \ store at addr in wcb seg

\ fetch word from addr in current wcb
: wcbseg@                       \ addr -- n
  wcbseg @ swap e@ ;            \ fetch from addr in wcb seg
```

```
file: WINDOW.BLK        Block: 12

\ window routines                                    cl 11/10/85
\ window frame drawing routines
: top
  ulx wcbseg@ uly wcbseg@ [ 201 boarder 256 * + ] literal putch
  ulx wcbseg@ 1+ uly wcbseg@ [ 205 boarder 256 * + ] literal
  width wcbseg@ draw_row
  ulx wcbseg@ width wcbseg@ + 1+ uly wcbseg@
  [ 187 boarder 256 * + ] literal putch ;
: bottom
  ulx wcbseg@ uly wcbseg@ height wcbseg@ + 1+
  [ 200 boarder 256 * + ] literal putch
  ulx wcbseg@ 1+ uly wcbseg@ height wcbseg@ + 1+
  [ 205 boarder 256 * + ] literal width wcbseg@ draw_row
  ulx wcbseg@ width wcbseg@ + 1+ uly wcbseg@ height wcbseg@ + 1+
  [ 188 boarder 256 * + ] literal putch ;
```

```
file: WINDOW.BLK        Block: 13

\ window routines                                    cl 11/10/85

\ window frame drawing routines
: sides
  uly wcbseg@ height wcbseg@ + 1+ uly wcbseg@ 1+
  do ulx wcbseg@ i [ 186 boarder 256 * + ] literal putch
     ulx wcbseg@ width wcbseg@ + 1+ i
```

```
    [ 186 boarder 256 * + ] literal putch
  loop ;
```

```
file: WINDOW.BLK       Block: 14

\ window routines                                      cl 11/10/85

\ temporary data storage areas
\ used by scn->buf and buf->scn

label save_h   nop nop          \ storage for height parameter
label save_w   nop nop          \ storage for width  parameter
label save_ptr nop nop          \ storage for start pointer
label save_si  nop nop          \ storage for forths IP reg
label save_ds  nop nop          \ storage for current ds reg
```

```
file: WINDOW.BLK       Block: 15

\ window routines                                      cl 11/10/85
\ move data from screen to memory buffer
hex
code scn->buf                    \ x y width height seg --
 cld es pop 0 # di mov save_h #) pop  save_w #) pop ax pop
 a0 # bl mov bl mul bx pop bx shl bx ax add ax save_ptr #) mov
 si save_si #) mov ds ax mov ax save_ds #) mov v_seg # ax mov
 ax ds mov cs: save_ptr #) si mov cs: save_h #) cx mov
 here cx push cs: save_w #) cx mov  rep movs
      cs: save_ptr #) si mov a0 # si add si cs: save_ptr #) mov
      cx pop
 loop
 cs: save_ds #) ax mov  ax ds mov
 save_si #) si mov
 next
end-code
```

```
file: WINDOW.BLK       Block: 16

\ window routines                                      cl 11/10/85
\ move data from memory buffer to screen
code buf->scn                    \ seg x y width height --
 cld  save_h #) pop  save_w #) pop ax pop  a0 # bl mov
 bl mul bx pop bx shl bx ax add ax save_ptr #) mov
 si save_si #) mov ds ax mov ax save_ds #) mov ax pop ax ds mov
 v_seg # ax mov ax es mov 0 # si mov cs: save_ptr #) di mov
 cs: save_h #) cx mov
 here cx push cs: save_w #) cx mov  rep movs
      cs: save_ptr #) di mov a0 # di add di cs: save_ptr #) mov
      cx pop
 loop
 cs: save_ds #) ax mov  ax ds mov   save_si #) si mov
 next
end-code
decimal
```

```
file: WINDOW.BLK       Block: 17

\ window routines                                      cl 11/10/85
\ lowest level window routine
```

(continued on next page)

NOT COPY PROTECTED!

Sybil is an Advanced Diagnostics disk . . .
She can low format hard disks just like Advanced Diagnostics (IBM, Compaq, etc.) and she can do system and memory tests which provide even more information than Advanced Diagnostics does. $245.00 cheaper than IBM's Advanced Diagnostics!

Sybil is a Disaster Recovery program . . .
She can recover hard disks that have been accidentally formatted, completely! The hard disk reappears in exactly the same condition prior to the format. Truly amazing!

Sybil is a Graphics Editor . . .
She can draw on either RGB monitors (in color) or IBM Monochrome monitors in high ASCII characters. Perfect for creating Binary Image Files. The Binary Image Files can be converted to Assembly and then linked to other languages, such as your favorite Pascal, C, or compiled BASIC program. Includes source code.

Sybil is a File Wizard . . .
Sybil can backup files by **date,** by **time,** or by **size**. She can find any file (or files) anywhere on your hard or floppy disks, even if you haven't the vaguest notion. She can edit file attributes with the greatest of ease, unerase files, edit sectors, and globally change time and date stamps. All her file utilities understand paths and wildcards.

Sybil is also a . . .
RAM Disk, Print Spooler, General Regular Expression Parser and, **Advanced File Comparator**.

Order Sybil Today!
Call 800-922-3001, in Colorado, 303-444-1542

MasterCard VISA AMERICAN EXPRESS

SOPHCO
PO Box 7430
Boulder, Colorado 80306

Circle **no. 298** on reader service card.

Listing One (Listing continued, text begins on page 46.)

```
\ moves screen data to memory buffer
\ and then draws the actual window frame

: ((window))                        \ move data scn->buf
  ulx wcbseg@ uly wcbseg@           \ x y coordinates
  width wcbseg@ 2+ height wcbseg@ 2+ \ width height
  bufseg wcbseg@ scn->buf           \ get buf seg addr
  top sides bottom ;
```

```
file: WINDOW.BLK        Block: 18

\ window routines                                  cl 11/10/85

\ clear window routine
: clr_window                        \ --
  ulx wcbseg@ 1+                    \ upper left  corner x
  uly wcbseg@ 1+                    \ upper right corner y
  ulx wcbseg@ width wcbseg@ +       \ lower left  corner x
  uly wcbseg@ height wcbseg@ +      \ lower right corner y
  0 attrib wcbseg@ scrlup           \ scroll entire window
  0 curx wcbseg!                    \ home window cursor
  0 cury wcbseg! ;
```

```
file: WINDOW.BLK        Block: 19

\ window routines                                  cl 11/10/85
: (window)                          \ x y width height attrib -- f
  record_size calloc                \ try to allocate space for wcb
  if wcbseg @ >r wcbseg ! r>        \ if successful store seg var
     oldwcbseg wcbseg! attrib wcbseg!   \ save attrib in wcb
     2dup 2+ swap 2+ * 2* calloc \ alloc space for screen buf
     if bufseg wcbseg!              \ save buffer seg
        height wcbseg! width wcbseg!    \ save parameters in
        uly    wcbseg! ulx   wcbseg!    \ new wcb
        rdcur oldy wcbseg! oldx wcbseg! \ get old cursor pos.
        ((window)) clr_window true      \ move data draw frame
     else ." buffer alloc. failure" cr  \ if no memory
       wcbseg @ free drop drop 0 \ free wcb memory
     then
  else ." wcb alloc. failure" drop drop 0
  then ;                                \ return flag
```

```
file: WINDOW.BLK        Block: 20

\ window routines                                  cl 11/10/85
\ window parameter checking
: wfit cr
  abort" Window won't fit on crt" ;
: open_window                       \ x y width height attrib -- f
  depth 5 >=
   if >r 4dup rot + 2+ 24 <=
     if + 2+ 79 <=
       if r> (window)
       else cr ." ULX and/or WIDTH incorrect" wfit
       then
     else    cr ." ULY and/or HEIGHT incorrect" wfit
     then
   else cr ." Incorrect # of parameters specified" quit
   then ;
```

```
file: WINDOW.BLK        Block: 21

\ window routines                                  cl 11/10/85
\ close the current window (defined by wcbseg data)
\ free wcb and buffer memory then unlink window
: close_window                      \ --
  wcbseg @ 0 <>                     \ if window exists
  if bufseg wcbseg@                 \ get buffer seg addr
     ulx wcbseg@ uly wcbseg@        \ get x,y corner
     width wcbseg@ 2+ height wcbseg@ 2+
     buf->scn                       \ mov data back to screen
     oldx wcbseg@ oldy wcbseg@ at
     bufseg wcbseg@ free drop       \ free buffer seg memory
     wcbseg @ free drop             \ free wcb seg memory
     oldwcbseg wcbseg@ wcbseg ! \ unlink this window
  else                              \ if no current window
     cr ." No open windows !" cr
  then ;
```

```
file: WINDOW.BLK        Block: 22

\ window routines                                  cl 11/10/85
\ position cursor in window
\ if parameters out of range do the best we can and still
\ stay in the window
: wat                               \ x y --
  swap dup abs width wcbseg@        \ req. x in window ?
  1- >                              \ if not then
  if drop width wcbseg@ 1- then     \ set x to max in window
  curx wcbseg!                      \ save new cursor x position
  dup abs height wcbseg@            \ req y in window ?
  1- >                              \ if not then
  if drop height wcbseg@ 1- then \ set y to max in window
  cury wcbseg!                      \ save new cursor y position
  curx wcbseg@ ulx wcbseg@ + 1+     \ actual cursor position
  cury wcbseg@ uly wcbseg@ + 1+     \ calculation
  at ;
```

```
file: WINDOW.BLK        Block: 23

\ window routines                                  cl 11/10/85
\ read window cursor position
: rdwcur                            \ -- x y
  curx wcbseg@ cury wcbseg@ ;

\ read char/attrib of character at cursor in window
```

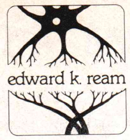
```
: rdwcha                              \ x y -- char/attrib
  wat rdchra ;

\ scroll window up for blank line at bottom
: scroll_window                       \ --
  ulx wcbseg@ 1+ uly wcbseg@ 1+ \ upper left corner to scroll
  ulx wcbseg@  width wcbseg@ +  \ lower right x coordinate
  uly wcbseg@ height wcbseg@ +  \ lower right y coordinate
  1 attrib wcbseg@ scrlup ;     \ up 1 line
```

```
file: WINDOW.BLK       Block: 24

\ window routines                                    cl 11/10/85
\ do carrage return in the current window
: crout rdwcur nip 0 swap wat ; \ carrage ret in window

\ do a line feed in the current window
: lfout rdwcur 1+ dup
  height wcbseg@ 1- >           \ cursor out of window
  if 1- scroll_window then      \ if so scroll the window up
  wat ;                         \ place the cursor in window

\ do a back space in the current window
: bsout rdwcur over 0<>         \ backspace cursor in window
  if swap 1- swap wat then ;

\ ring the bell
: bell 7 (emit) ;               \ sound the horn
```

```
file: WINDOW.BLK       Block: 25

\ window routines                                    cl 11/10/85
: wemit dup 32 <                \ char --
  if case                       \ if control char process it
       7 of bell   endof        \ if bell then
       8 of bsout endof         \ if backspace then
      10 of lfout endof         \ if linefeed then
      13 of crout endof         \ if carrage ret then
     endcase
  else                          \ else its a display char
     attrib wcbseg@ 256 * +     \ char now char/attrib
     rdwcur rot chra+           \ output char adv. cursor
     drop dup width wcbseg@ 1- -   \ if at end of window line
     if drop lfout crout        \ do lfcr to next line
     else 1+ curx wcbseg!       \ store new x coordinate
     then
  then ;
```

```
file: WINDOW.BLK       Block: 26

\ window routines                                    cl 11/10/85
: wcr 13 wemit 10 wemit ;       \ window carrage return

: wtype 0                       \ window equiv. of type
  ?do count wemit loop drop ;

\ use memory manager to give forth a full 64k segment
```

(continued on next page)



Transform Your Programs with CPP—C Preprocessor Plus

Includes ALL features of the standard C preprocessor.
- Define arbitrarily complex macros with #define command.
- Include and nest files to any depth with #include command.
- Include lines with #if, #ifdef and #ifndef commands.
- Define multiple constants with #enum command.
- Optional extra feature: Imbed formatting or other commands in your source code. (Lines starting with . or * are ignored.)

Fast and flexible
- 30 times faster than the Preprocessor published in Dr. Dobb's Journal.
- Can be used for any language, including assembler.
- Can be used as a stand-alone macro/include processor.
- Code can be used as the lexical analyzer for parsers or assemblers.

Complete
- You get complete SOURCE CODE in standard C.
- You get everything you need to use CPP immediately.
- CPP is unconditionally guaranteed. If for any reason you are not satisfied with CPP, your money will be refunded promptly.

Price: $95. Call or write today:
Edward K. Ream
1850 Summit Ave., Dept. DD
Madison, WI 53705
(608) 231-2952

TO ORDER: Specify both the operating system (MS-DOS, CP/M 80 or CPM 68K) and the disk format (8 inch CP/M or the exact type of 5¼ inch disk). Send a check or money order for $95 ($105 for foreign orders). Foreign checks must be denominated in U.S. dollars drawn on a U.S. bank. Sorry, I do NOT accept phone, credit card or COD orders. Please do NOT send purchase orders unless a check is included.

Circle **no. 90** on reader service card.

Disclone Service Won't Make You Nervous

Technical support, personal service, competitive prices.
Disclone full service quality tested diskette duplication, packaging, documentation production and processing ensures precise duplication, thorough quality control, and expedient response to your requirements.
NOclone state of the art hardware based copy protection is true piracy protection for authorized allotments only. Each application is uniquely encrypted. Install routines are coded for nontransferrable hard disk allotments.
Committment dates are guaranteed. Fast turnover
- up to 1000 in 24 hours, any format.
- up to 10,000 in one week, any format.

disclone

DISCLONE SOFTWARE PRODUCTION SERVICES
1050 North 5th street, San Jose, California 95112
(408) 947-1161 OUTSIDE CA: 1-800-826-4296

Circle **no. 204** on reader service card.

FORTRAN PROGRAMMERS

Looking for the right PC FORTRAN LANGUAGE SYSTEM? If you're serious about your FORTRAN programming then you should be using F77L- LAHEY FORTRAN.

Editor's Choice - PC Magazine

- Full FORTRAN 77 Standard (F77L is not a subset)
- Popular Extensions for easy porting of minicomputer and mainframe applications
- COMPLEX*16, LOGICAL*1 and INTEGER*2
- Recursion - allocates local variables on the stack
- IEEE - Standard Floating Point Arithmetic
- IMPLICIT NONE
- Long variable names - 31 characters
- Fast Compile - Increase your productivity
- Source on Line Debugger (Advanced features without recompiling)
- Arrays and Common Blocks greater than 64K
- Clear and Precise English Diagnostics
- Compatibility with Popular 3rd Party Software (i.e. Lattice C)
- Easy to use manual
- Technical Support from LCS

F77L - THE PROGRAMMER'S FORTRAN

$477.00 U.S.

System Requirements: MS-DOS or PC-DOS, 256K, math coprocessor (8087/80287)

FOR MORE INFORMATION: (702) 831-2500

Lahey Computer Systems Inc.
P.O. Box 6091 Incline Village, NV 89450/U.S.A.

International Dealers:

England:	Grey Matter Ltd.,	Tel: (0364) 53499
Denmark:	Ravenholm Computing,	Tel: (02) 887249
Australia:	Computer Transitions	Tel: (03) 537-2786
Japan:	Microsoftware, Inc.,	Tel: (03) 813-8222

SERVING THE FORTRAN COMMUNITY SINCE 1967

Circle **no. 186** on reader service card.

THE HAMMER
Software Tools in C

"I have already saved weeks of coding . . . thank you for providing such a useful tool . . ." - G.T.

Let **The HAMMER Library** of over 150 routines save you valuable development time and effort:

LIBRARY FUNCTIONS
- **Multi-level 123-like MENUS**
- **DATA ENTRY**
  - MULTI-FIELD mode for Full-Screen data entry
  - Single-Field mode for individual fields
  - Data Verification
  - Full Editing within each field
  - Strings, dates, and fixed decimal numbers
  - "Option" fields force user to pick from a given set
- **SCREEN MANAGEMENT**
  - cursor positioning
  - full attribute control
  - display boxes & tables
  - scrolling and clearing
- Date/time/string conversions
- BIOS access/pattern matching/and more

UTILITIES
- HARC - complete Source File Archiver
- HAMCC - compile designated source modules residing **WITHIN an archive file** under any of the supported compilers and optionally place resulting object modules in a library.

SUPPORTED C COMPILERS:
Microsoft C 3.00 • CI-C86 • Mark Williams C86
DeSmet C • Lattice

INCLUDES source code and manual

$195 plus shipping
VISA/MC accepted

O.E.S. SYSTEMS
1906 Brushcliff Rd. • Pittsburgh, PA 15221 • 412/243-7365

Looking for the right tool for the job?
REACH FOR THE HAMMER

Circle **no. 137** on reader service card.

FORTH WINDOWS

Listing One

(Listing continued, text begins on page 46.)

```
: initialize                            \ --
  cr ." Memory management "             \ output 1/2 msg
  -1 setblock                           \ request FFFF bytes
  if                                    \ if successful
    ." initialized"                     \ output message and
    0 wcbseg !                          \ initialize link variable
  else
    ." error" quit                      \ abort program
  then cr ;
```

```
file: WINDOW.BLK        Block: 27

\ window demo                                              cl 11/10/85
\ window equivalents of standard Forth words

: wlist block 16 0
  do dup i c/l * + c/l                  \ window equiv. of list
     -trailing wtype wcr
  loop drop ;

: wtriad 3 / 3 * 3 bounds               \ window equiv. of triad
  do i wlist                            \ list screen in window
     wcr wcr                            \ add a couple of cr's
  loop ;
```

```
file: WINDOW.BLK        Block: 28

\ window demo                                              cl 11/10/85
\ window canned messages
: msg1
  " This could be your application program! " wtype ;
: msg2 " Ain't this window package something! " wtype ;
: msg3 "  ** Window 4 ** " wtype ;

: msg1out 0 0 wat                       \ output msg1 20 times
  20 0 do msg1 loop ;

: msg2out 0 0 wat                       \ output msg2 10 times
  10 0 do msg2 loop ;

: msg3out 0 0 wat                       \ output msg3 80 times
  80 0 do msg3 loop ;
```

```
file: WINDOW.BLK        Block: 29

\ window demo                                              cl 11/10/85
\ video attribute constants
7   constant normal          15 constant high_int
112 constant reverse         128 constant blink

: fill_crt 0 0                          \ fill crt with rev video A's
  [ ascii A reverse 256 * + ]           \ calculate char/attrib code
```

```
  literal 2048 draw_row ;

: wait 10000 0 do noop loop ;    \ timing loop
```

```
file: WINDOW.BLK        Block: 30

\ window demo                                           cl 11/10/85
\ define the four windows used in the demo program

: window1                        \ define window #1
  0 0 20 10 reverse open_window ;

: window2                        \ define window #2
  2 1 70 8 normal open_window ;

: window3                        \ define window #3
  7 6 69 10 reverse open_window ;

: window4                        \ define window #4
  10 9 59 4 high_int open_window ;
```

```
file: WINDOW.BLK        Block: 31

\ window demo                                           cl 11/10/85
: demo
  fill_crt window1
 if 0 0 wat msg2 wait wcr wait 7 emit wcr
    wait " It sure is" wtype wait 8 wemit 8 wemit
    wait 10 5 wat wait window2
    if msg1out wait window3
       if 0 10 wat 24 wtriad wait window4
          if msg3out wait close_window wait close_window
             wait clr_window msg2out wait close_window
             0 wlist wait wait wait wait close_window
          then
       then
    then
 then
 wait ;
```

```
file: WINDOW.BLK        Block: 32

\ window demo                                           cl 11/10/85
only forth also dos also         \ search dos and forth
: test empty-buffers             \ dummy program name
  initialize                     \ initialize memory manager
  " window.blk"  fcb1 (!fcb)     \ parse filename to fcb
  fcb1 !files open-file          \ open the file to list
  2 0
  do                             \ run the demo 2 times
    demo wait wait wait dark wait
  loop
  ." What did you think of that Huh?" cr bye ;

only forth also                  \ power up search order

' test is boot                   \ make demo run automatically
save-system window.com           \ create .COM demo
```

End Listing

nine track tape users
Micro to Mainframe Connection

The Model TC-50 ½-inch tape subsystem provides a standard medium for transmission of mainframe data base information to PC users, while maintaining mainframe isolation and data integrity. Use ODI subsystems to import data to data base programs like dBase III.

The TC-50 subsystem also provides fast back-up capability as well as a device driver and interface software for popular compilers.

The TC-50 subsystem includes tape drive controller, cables and documentation. All ODI products carry a 30 day unconditional money-back guarantee, and are warranted for one year, parts and labor.

ODI

Overland Data, Inc.
5644 Kearny Mesa Road
San Diego, CA 92111
Tel. (619) 571-5555
Telex 754923 OVERLAND

Also Available – XENIX tape subsystems for the IBM AT

Circle **no. 192** on reader service card.

¿C? ¡Sí!

If you're a C programmer (or want to be one), we speak your language. Subscribe to **The C Journal** today, and start increasing your productivity right away. We give you information you can **use** on **any** machine — IBM PC™, UNIX™-based, Macintosh™, or CP/M™ — micro, mini, or mainframe.

- in-depth reviews and feature articles — C compilers, editors, interpreters, function libraries, and books.
- hints and tips — help you work **better** and **faster**.
- interviews — with software entrepreneurs that **made it** — by using C.
- news and rumors — from the ANSI standards committee and the industry.

Limited Time Offer

Join our thousands of subscribers at the **Discount Rate** of only $18 for a full year (regularly $28)! Call us now at (201) 989-0570 for faster service — don't miss a single issue of **The C Journal!**

Please add $9 for overseas airmail.

Trademarks — CP/M: Digital Research Inc. IBM PC: IBM Corp. Macintosh: Apple Computer Corp. **The C Journal:** InfoPro Systems. UNIX: AT&T Bell Labs.

InfoPro Systems
3108 Route 10
Denville, NJ 07834
(201) 989-0570

MasterCard VISA

Circle **no. 194** on reader service card.

Listing One (Text begins on page 112.)

```
Listing One:  tiny tools

: NIP   ( n m - m )        SWAP DROP ;   ( drops second on stack  )
: TUCK  ( n m - m n m )    SWAP OVER ;   ( tucks top under second )
: -ROT  ( a b c - c a b )  ROT ROT ;     ( opposite of ROT )

: INCR  ( a - )   1 SWAP +! ;    ( increments a variable )
: DECR  ( a - )  -1 SWAP +! ;    ( decrements a varaible )

  ( ERRCNT INCR increments the variable ERRCNT )
  ( #LINES DECR decrements the variable #LINES )

: ON    ( a - )  -1 SWAP ! ;   ( forces variable to true value )
: OFF   ( a - )   0 SWAP ! ;   ( forces variable to false value )

   27 CONSTANT ESC

: NUF? ( - f ) ?TERMINAL DUP IF KEY 2DROP KEY ESC = THEN ;

  ( NUF? is used inside a DO LOOP structure; when a key is   )
  ( pressed, NUF? stops to wait for a second keypress.        )
  ( If no key was struck or the second key is not the Escape  )
  ( key, the flag is false; otherwise, the flag is true.      )

  ( The word ?TERMINAL is vendor-specific.  Your Forth might  )
  ( use ?KEY or some other word instead.  It is the word that )
  ( returns a true flag if a key has been pressed and a false )
  ( flag otherwise; the key's value is then retrieved with the)
  ( standard word KEY. )

: LISTIT  #LINES 0 DO I LIST-LINE NUF? IF LEAVE THEN LOOP ;

 ( Example of NUF?:  LISTIT will pause after listing a line )
 ( when a key is pressed.  Pressing Escape will then leave  )
 ( the loop, interrupting the task; any key except Escape   )
 ( does not interrupt but resumes the task. )

   0 CONSTANT F
  -1 CONSTANT T

: ESC-HIT? ( - f ) ( leaves T if Escape key pressed )
   F ?TERMINAL IF BEGIN KEY ESC = OR ?TERMINAL NOT UNTIL THEN ;

 ( ESC-HIT? is like NUF? without the pause.  It is used much )
 ( like NUF? and would also work in the above example.  Note )
 ( that ESC-HIT? discards the contents of the key buffer as  )
 ( it rummages through looking for an Escape keypress.       )

: BYTE-SWAP ( x - x' ) 256 UM* OR ;

 ( This word swaps the two bytes in a cell.  Bill Muench of  )
 ( Santa Cruz thought of this little gem. )
```

End Listing One

Listing Two

```
Listing Two: array-defining word 1

: ARRAY  CREATE ( # - ) 2* ALLOT    ( reserves # cells in memory )
         DOES>  ( n <adr> - adr ) SWAP 2* + ; ( adr of nth cell )

( This array allocates the number of cells specified, but does )
( not initialize them to zero. )

  8 ARRAY TOM    ( defines TOM as having 8 cells = 16 bytes )
  125 5 TOM !    ( stores 125 in cell 5 of TOM )
  0 TOM @        ( retrieves the contents of cell 0 of TOM )
```

End Listing Two

Listing Three

```
Listing Three:  array-defining word 2

  1 CONSTANT BYTES
  2 CONSTANT CELLS
  4 CONSTANT DOUBLES

: FOR   CREATE ( #slots type - ) DUP C, * HERE OVER ERASE ALLOT
        DOES>  ( index <adr>  - adr ) COUNT ROT * + ;

  11 BYTES FOR FRED
  35 CELLS FOR JOAN
  17 DOUBLES FOR JOHN

 ( These arrays will deliver the address of the slot based  )
 ( on the type of the entry.  The array is initialized to   )
 ( zeroes at creation time.  It is the programmer's job to  )
 ( use C!, !, 2!, C@, @, and 2@ as appropriate.  Note that  )
 ( FRED's 11 slots are numbered 0 through 10, JOAN's 35 are )
 ( numbered 0 through 34, and JOHN's 17 are 0 through 16.   )

  213 3 FRED C!         ( stores 213 into byte 3 of FRED )
  31 JOAN @             ( fetches contents of cell 31 of JOAN )
  3142352. 15 JOHN 2!   ( stores 3142352. into slot 15 of JOHN )
```

End Listing Three

Listing Four

```
Listing Four:  array-defining word 3

  1 CONSTANT PUT    ( flags for the IF statement )
  0 CONSTANT GET    ( in the DOES> part of FOR   )

  CREATE STORES     ] C! ! NOOP 2! [   ( NOOP stored to put 2! )
  CREATE FETCHES    ] C@ @ NOOP 2@ [   ( and 2@ in right spot  )

: FOR   CREATE ( #slots type - ) DUP C, * HERE OVER ERASE ALLOT
        DOES>  ( datum 1 ndx <adr> --   |   0 ndx <adr> -- datum )
           COUNT DUP >R ( save type ) ROT * +   R> 1- 2*   ROT
              IF STORES  ELSE FETCHES  THEN  + @ EXECUTE ;

 ( This version of FOR takes care of the fetching and storing  )
 ( given the appropriate flag; the programmer does not have to )
 ( remember whether it is a byte, cell, or double-precision    )
 ( array.  This could easily be extended for floating-point    )
 ( numbers as well. In the stack comment, "|" is read as "or." )

  11 BYTES FOR FRED
  35 CELLS FOR JOAN
  17 DOUBLES FOR JOHN

  213 PUT 3 FRED         ( stores 213 in byte 3 of FRED )
  GET 31 JOAN            ( retrieves contents of cell 31 of JOAN )
  3142352. PUT 15 JOHN   ( stores 3142352. in slot 15 of JOHN )
```

End Listing Four

Listing Five

```
Listing Five:  bit tools

  CREATE BITBYTES  1 C, 2 C, 4 C, 8 C, 16 C, 32 C, 64 C, 128 C,

: FLAG  ( ? - f ) 0= NOT ; ( forces to a Boolean flag: -1 or 0 )

: AIM   ( # adr - bit# adr' ) SWAP 8 /MOD ROT + ;

: +BIT  ( # adr - ) AIM SWAP MASK OVER C@  OR SWAP C! ;

: -BIT  ( # adr - ) AIM SWAP MASK NOT OVER C@ AND SWAP C! ;

: @BIT  ( # adr - f ) AIM C@ SWAP MASK AND FLAG ;

: ~BIT  ( # adr - f ) AIM 2DUP @BIT IF -BIT ELSE +BIT THEN ;
```

End Listing Five

Listing Six

```
Listing Six:  array-defining word 4

  0 CONSTANT BITS        ( for bit arrays )

: BITS>BYTES  ( #bits - #bytes ) 8 /MOD SWAP IF 1+ THEN ;

: FOR    CREATE ( #slots type - ) DUP C, ?DUP
            IF *   ELSE BITS>BYTES   THEN
            HERE OVER ERASE ALLOT
         DOES> ( datum 1 ndx <adr> --   |   0 ndx <adr> -- datum )
            COUNT ?DUP   ( nonzero = numbers; 0 = bits )
            IF DUP >R ( save type )  ROT * +  R> 1- 2*  ROT
               IF STORES  ELSE FETCHES  THEN  + @ EXECUTE
            ELSE  ROT ( action flag: 1 = store, 0 = fetch  )
               IF ROT ?DUP  ( nonzero means 1 bit or toggle )
                  IF 0<  IF ~BIT  ELSE +BIT THEN
                  ELSE -BIT THEN
               ELSE @BIT THEN THEN ;

  1 1   2CONSTANT SET     ( By placing two values on  )
  0 1   2CONSTANT ZAP     ( the stack, these words in )
 -1 1   2CONSTANT FLIP    ( effect include the PUT.   )

 23 BITS FOR BIT     ( reserves 4 bytes for bit array )

  SET 16 BIT     ( turns bit 16 on )
  ZAP  5 BIT     ( turns bit 5 off )
  FLIP 0 BIT     ( toggles bit 0    )

  GET 3 BIT      ( retrieve bit 3 as boolean flag )

 ( Examples shown in Listing 4 will also work with this word.)
```

End Listing Six

Listing Seven

```
Listing Seven:  array-defining word 5

: >TYPE  ( adr - adr' ; from #slots-adr to type-adr )  2+ ;

: >DATA  ( adr - adr' ; from #slots-adr to data-adr )  3 + ;

: FOR   CREATE ( #slots type - )
              OVER , ( #slots )   DUP C, ( type )  ?DUP
              IF *   ELSE BITS>BYTES   THEN
              HERE OVER ERASE ALLOT
        DOES> ( datum 1 ndx <adr> --  |  0 ndx <adr> -- datum )
              >TYPE COUNT ?DUP ( nonzero = numbers; 0 = bits )
              IF DUP >R ( save size ) ROT * + R> 1- 2* ROT
                 IF STORES  ELSE FETCHES  THEN  + @ EXECUTE
              ELSE  ROT ( action flag:  1 = store, 0 = fetch )
                 IF ROT ?DUP ( nonzero means 1 bit or toggle )
                    IF 0< IF ~BIT ELSE +BIT THEN
                    ELSE -BIT THEN
                 ELSE @BIT THEN THEN ;
```

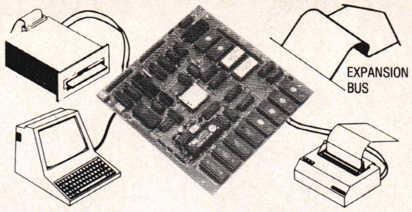
End Listing Seven

Listing Eight

```
Listing Eight:  array display word

: "TYPES ." bit   byte  cell        double" ;

: .TYPE ( type - ) 6 * ['] "TYPES >BODY 3 + + 6 -TRAILING TYPE ;

: DOUBLE? ( type - f ) 4 = ;

: }LINE   ( type n - type ) OVER DOUBLE?  IF DUP 5  ELSE DUP 10
    THEN   MOD  IF DROP  ELSE CR 4 .R ."  | " THEN ;

: VITALS ( array-adr - data-adr #slots type ) DUP >TYPE
    OVER >DATA ROT @ ( #slots ) ROT C@ ( type ) ;

: TITLE  ( #slots type - ) CR CR SWAP . .TYPE ." s:" ;

: DISPLAY ( adr -- )  VITALS 2DUP TITLE  ?DUP
    IF ( numbers ) SWAP 0 DO I }LINE 2DUP I * + ( adr )
        OVER DUP >R ( save type ) 1- 2* FETCHES + @ EXECUTE
        R> DOUBLE?   IF 12 D.R   ELSE 7 .R   THEN
        NUF? IF LEAVE THEN LOOP 2DROP
    ELSE ( bits ) 0 DO I DUP }LINE OVER @BIT
        2 SPACES  IF ASCII 1  ELSE ASCII -  THEN EMIT
        NUF? IF LEAVE THEN LOOP DROP
    THEN CR ;

: SPILL ( - ; name ) BL WORD FIND
   IF   >BODY DISPLAY
   ELSE DROP CR ." No such array " THEN ;


16 DOUBLES FOR MIKE
1892735. PUT  0 MIKE
7802472. PUT 15 MIKE
1263.    PUT  8 MIKE
SPILL MIKE

16 doubles:
  0 |      1892735           0           0           0           0
  5 |            0           0           0        1263           0
 10 |            0           0           0           0           0
 15 |      7802472

16 BITS FOR STEVE
SET  0 STEVE
SET 15 STEVE
FLIP 11 STEVE
SPILL STEVE

16 bits:
  0 |  1  -  -  -  -  -  -  -  -  -
 10 |  -  1  -  -  -  -  1
```

End Listings

OEM188 SBC DEVELOPMENT SYSTEM FOR PRODUCT APPLICATIONS



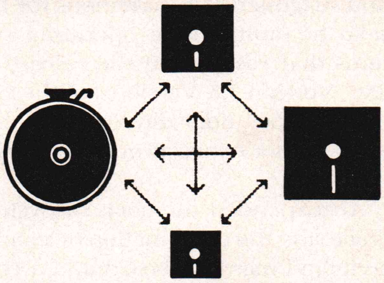
The OEM188 - designed to bring your product to market in the fastest possible time - through the most productive software development environment available & cost effective hardware.

- The OEM188 boots MS-DOS or CP/M-86. Write your program in Assembler, Forth, Basic, C, Fortran or Pascal.
- ROM your code. The EPROM programmer is onboard and fully integrated into the hardware and software.
- Develop your code quickly with Vesta's ROMmed languages designed for control tasks.

Size 8" x 8". FDC for 4 drives, Dual UART with RS-232, TTL and RS-422 I/O, Bus - IBM, Printer port, Watchdog, Battery backed real time clock and up to 256 K static RAM/ROM. Programmer interface - terminal. Various I/O boards available.
Prices starting as low as $329 each
VESTA TECHNOLOGY, INC.•7100 W. 44th Ave.•Suite 101
Wheatridge, CO 80033 • (303)422-8088 • VISA & MC

Circle **no. 198** on reader service card.

DATA CONVERSION

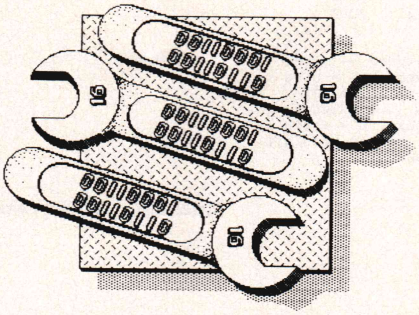


TRANSFER DATA BETWEEN OVER 500 DIFFERENT COMPUTER SYSTEMS
WORD PROCESSORS TOO
QUICK TURN-AROUND
PRICES FROM $9 PER DISK
CALL OR WRITE FOR YOUR
FREE CATALOG
PORT-A-SOFT
555 S. STATE ST., SUITE #12
P.O. BOX 1685, OREM, UT 84057
(801) 226-6704

Circle **no. 229** on reader service card.

Choose your weapon ...
ForthCard
or
C Board
Our single board computers allow you to write high level embedded applications.
HiTech Equipment Corporation
9560 Black Mountain Road
San Diego, CA 92126
(619) 566-1892

Circle **no. 196** on reader service card.



by Ray Duncan

Forth and the EMS

The Lotus/Intel/Microsoft Expanded Memory Specification (EMS) allows application programs to access as much as 8 megabytes of bank-switched memory in a portable manner. The original specification (Version 3.0) was released jointly by Intel and Lotus around the time of the spring Comdex in 1985. Subsequently, Microsoft endorsed the EMS and suggested some additions for the sake of multitasking operating systems that resulted in the release of EMS, Version 3.2. Version 4 of the EMS is said to be under development, but we will not concern ourselves with that here.

An expanded memory subsystem is actually the combination of a bank-switched memory board and a resident system driver. Taken together, these present a uniform interface that can be called by programs via a software interrupt (*67H*). The driver supports such functions as allocation, deallocation, and mapping of the expanded memory pages. The Above Board from Intel was the first commercially available implementation of the EMS, but EMS boards are now available from a broad variety of vendors, including AST, Quadram, and Tall Tree Systems. Expanded memory should not be confused with *extended memory*, which is IBM's term for the physical memory above 1 megabyte that is addressable by an 80286 CPU running in protected mode.

In harmony with this *DDJ* issue's emphasis on Forth, I have included the source code for a simple PC/Forth interface to an expanded memory subsystem, allowing the declaration and use of huge arrays (Listing One, next month). I have kept the code simple for clarity and have made no attempt to optimize it for speed. Also, for the purposes of this example, I have not taken advantage of the EMS driver's ability to map four separate logical pages onto physical memory at a time, and I have not included code to eliminate redundant mapping calls. As they say in the calculus books, these enhancements are left as an exercise for the reader, though they are easy to add once the basic EMS scheme is understood.

Readers wishing for more details about EMS programming can find them in the chapter on memory management in my book *Advanced MS-DOS* (Bellevue, WA: Microsoft Press, 1986). You can also obtain the original EMS document (part number 300275-003, dated September 1985) from Intel Corp., 3065 Bowers Ave., Santa Clara, CA 95051.

The TEE Filter Again

Herb Shear, of Los Altos, California, wrote: "Just a couple of comments regarding the TEE filter (published in the April 1985 *DDJ*). In the August issue you mention a bug reported by Dr. Fred Sinal, who indicated that the problem occurred with files that did not end in a *CR/LF*. The actual problem is with files ending with the infamous Control-Z being output to the console in cooked mode. The resultant stripping of the ^Z leaves the count one short, implying a 'disk full' to the calling program.

"My first fix was to use the returned count as an offset pointer into the buffer and to test the character for ^Z. If the compare was true, it was considered a good excuse for the failed write and the jump to the error message was not executed. This was OK, except that with midfile ^Zs, the latter part of the file never got to the console, though a file large enough to reload the buffer would cause console output to resume. The disk file output would be complete. Acceptable but rather ungainly behavior.

"My second fix was to test the device attribute word for the output device with the *IOCTL* function and to set the output device to raw mode if it was found to be the console. Because the console's mode is remembered program to program, it must be restored before TEE exits."

Gary Woodman, of Darwin, Australia, wrote: "Using PC-DOS, I normally redirect the output from BACKUP into a file to keep a log of what was backed up and when. BACKUP is even duller than usual when there's no output on the screen, however.

"I gleefully recalled the TEE filter you published in the April 1985 issue and chuckled to myself: 'Ah ha, I'll TEE the output of BACKUP both to a file and to the screen!' But when I dug out the issue, it seemed like too much trouble to type in a couple of pages of 8086 assembler (it usually does), so I scratched my head for a few moments and came up with a short C program [Table 1, below].

"Now I know this is not quite the same as that Mr. Head provided, and I don't want to make a big thing of this, but it seems that the contrast between Mr. Head's MASM program and this C program represents, in a nutshell, the dichotomy of programming today. As well as contrasting

```
#include <stdio.h>
main( )
{       int c;
        while ( (c  =  fgetc(stdin) )  !=
                                        EOF){
                fputc(c,  stdout);
                fputc(c,  stderr);
        }
}
```

Table 1: *C version of TEE filter*

the source code, contrast Mr. Head's hours (possibly days) of development plus an hour of my time to type in and debug TEE.ASM (that is, debug my typing!) with the few minutes it took me to write the C program. . . . Could it be that we have here a case of the tail wagging the dog?

"Just in passing, in case anyone still feels that high-level language compilers add flab to our programs but little else, I report the code sizes of TEE.EXE as generated by my three MS-DOS compilers:

Computer Innovations, V. 1.31	7,936 bytes
DeSmet, V. 2.41	8,192 bytes
Hi-Tech, V. 2.0	1,611 bytes

"Incidentally, this version of the CI compiler is now in the public domain, a brilliant marketing ploy and a 'best buy.'

"I haven't done any benchmarks as it really isn't a very important issue. Everyone knows the assembler program leads by at least an order of magnitude. For the number of times I TEE things, I'm quite happy to accept the run-time inefficiencies in exchange for the saving in development using C for what is almost a disposable program. And as for TEEing the output of BACKUP, well, MS-DOS is of course single-tasking and, as it turns out, the output of BACKUP is not available to TEE until BACKUP finishes."

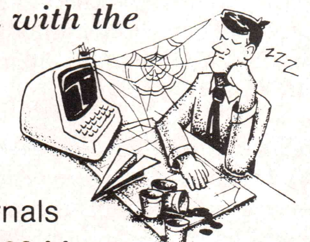
Mr. Woodman's points on coding time vs. execution time are certainly worth discussing further in this column. It doesn't take me more than an hour or two to write a program the size of TEE from scratch in assembly language and feel confident that it is adequately debugged. And the vast majority of assembly code is reusable, just as is C or any other language. Once an assembly-language functioning filter such as TEE is in hand, for example, it is only a few minutes' work to modify it to perform any reasonable transformation on a character stream. For me, the benefits of the superior performance and compactness of an assembly-language program almost always outweigh all other considerations for utility programs that I am going to run more than once. Let's hear from *DDJ* readers on this subject!

Incidentally, I suspect that if Unix

Relocating Macro Assemblers

•Z80 •8085 •NSC800 •HD64180

What is your turn-around time? Eliminate this dilemma with the fastest most powerful assemblers on the market.



- 32 significant chars. on externals
- math on externals
- T-states in listing
- local labels
- full drive-user support
- M80 Macros
- M80 Pseudo-ops
- tables overflow to disk
- recommend JR over JP

Requires Z80 W/40K TPA, CP/M2.2 or greater

$195

SLR Systems
1622 N. Main St., Butler, PA 16001
(800) 833-3061 (412) 282-0864
Telex 559215 SLR SYS

Circle **no. 78** on reader service card.

MS-DOS, UNIX, APPLE MAC, CP/M, NETWORKS and MORE. ONE c-tree ISAM DOES THEM ALL!

The creator of Access Manager™ brings you the most powerful C source code, B+ Tree file handler: c-tree™

- multi-key ISAM and low-level B+ Tree routines
- complete C source code written to K&R standards
- single-user, network and multi-tasking capabilities
- fixed and variable record length data files
- virtually opened files accommodate limited file descriptors
- no royalties on application programs

$395 COMPLETE

Specify diskette format:
- 5¼" MS-DOS
- 8" CP/M
- 3½" Mac
- 8" RT-11

For VISA, MC and COD orders call (314) 445-6833
FairCom
2606 Johnson Drive
Columbia, MO 65203

© 1985 FairCom

The following are trademarks: c-tree and the circular disk logo—FairCom; MS—Microsoft Inc.; CP/M and Access Manager—Digital Research Inc.; Unix—AT&T; Apple—Apple Computer Co.

Circle **no. 93** on reader service card.

(continued from page 107)

were written in assembly language instead of in C, it would have an overhead of 64K instead of 256K, would run ten times faster, and wouldn't be so overburdened with barely useful features that have been grafted on by generations of graduate-student programmers at UC Berkeley.

Defenders of Concurrent DOS

My (perhaps somewhat overdrawn) editorializations about Concurrent DOS and Digital Research in the February 1986 issue of *DDJ* drew plenty of responses from readers. This month I'd like to give some space to other viewpoints on this product.

Steve Elias, of Wellesley Hills, Massachusetts, wrote heatedly: "Look before you write that Concurrent DOS only supports DOS 1 functionality. You were using an old version. Concurrent DOS supports all 2.1 calls.

"Concurrent DOS is the most sophisticated and DOS-compatible multitasking operating system available. The fact that it is based on CP/M may be ugly, but it is transparent to the average user."

Mark Davidson of Chattanooga, Tennessee, sent two separate, detailed letters on Concurrent DOS 4.1, which I have abstracted to some extent: "The new version really only supports path names as far as the *CHDIR*, *RMDIR*, and *MKDIR* commands go. None of the Concurrent PC-DOS commands will accept a directory path. . . . As for using your DOS utilities under Concurrent DOS, it depends on what version of DOS these utilities belong to. If you are using PC-DOS 3.0 when you install Concurrent DOS, you can forget about running any of your PC-DOS utilities. They all refuse to run, giving the 'incorrect DOS version' message. The story is a little different if you are using PC-DOS 2.1. Some utilities will run fine (such as TREE and FIND), whereas others do things that are very unexpected (CHKDSK checks drive A: instead of the current drive). DRI warns that BACKUP and RESTORE should not be used. Also, it claims that you can run BASICA under Concurrent DOS with the only side effect being that multiuser activity will halt. This appears to be an understatement. BASICA on the PC/AT using the BASICA that comes with PC-DOS 3.1 will produce the 'incorrect DOS version' error. My attempts to run BASICA from PC-DOS 2.1 under Concurrent DOS produced an interesting display of nothing while causing a horrendous beeping to come from the speaker.

"I found some other unexpected surprises, too. Commands such as *type foo.doc ¦ more* will type the file but not pause at the end of each screenful. Could this mean that pipes don't work? Also, because none of the PC-DOS 3 commands will run under Concurrent DOS, attempts to execute the Version 3 command.com as a subprocess will fail. I haven't tried this with the PC-DOS 2.1 command.com.

"But this letter isn't full of just bad news. DRI has obviously fixed some of the problems that were present in Concurrent DOS 3.2. Many programs that wouldn't execute under 3.2 now run fine under 4.1—these include the Lattice C compiler and DeSmet's C compiler. Also, because some of the PC-DOS 2.1 utilities at least try to run, it is evident that several internal changes have been made. . . .

"Concurrent DOS is still noticeably slower than PC-DOS, even if you have only one task running. And it is a big system, still requiring approximately 1,700K of disk space; DRI also suggests that you have at least 512K RAM in your machine."

Brian J. Mullan of Lutz, Florida, sent an articulate defense of Concurrent DOS and its capabilities (both present and future). He wrote: "I am a senior software systems analyst with a company based in McLean, Virginia, and I would like to provide some comments regarding your editorial statements on Digital Research's Concurrent DOS operating system.

"IBM has just implemented DRI's Concurrent DOS 286 as the host IBM PC/AT OS for a 128-terminal system that it is going to market under the system 4680 name. I don't know how much you read in other professional trade magazines, but the IBM representatives who commented on the release of the 4680 system stated in an *MIS Week* issue three or four weeks ago that 'this is a true multitasking/multiuser operating system allowing IBM PC software to run in the IBM PC/ATs protected mode.'

"IBM also stated that it had worked with DRI to develop Concurrent DOS 286 and to allow it to run PC-DOS software in the AT's protected mode! The single AT running this OS not only provides the horsepower to manage the 128 dumb terminals tied to it (note: this system is not a LAN!), but the multitasking capabilities of the system allow on-line communications to an IBM mainframe host concurrently with other processing functions.

"So first of all, it is not true that DRI gave up on the 80286 Protected Mode version of its OS. Where is Microsoft's? [I didn't say DRI had given up on it, I said it admitted the OS would not be delivered in the form originally advertised.]

"The mention you made of the DRI Unix events are not quite accurate either. . . . According to the trade media coverage (by *Computerworld* and *MIS Week*), AT&T was the cause of the breakoff of the DRI Unix library development effort. Apparently AT&T decided that the Unix library needs could not possibly be fullfilled by a single company (which I think you should note included AT&T Informations Systems division) in the time necessary to bring AT&T's computer line up against IBM. Because AT&T had signed an exclusive agreement with DRI for the development of this library, this prohibited AT&T from parceling out the tremendous amount of work to be done to even a single other company. So it was at AT&T's request that the contract was canceled and with the acknowledgement that DRI was not delinquent in its contractual obligations to AT&T.

"Again, AT&T did not go to Microsoft for this development work, which should also make some sort of a statement!

"Last, Digital Research's Concurrent DOS version for the IBM PC/XT is the only true multitasking/multiuser, PC-DOS-compatible OS for those machines. Yes, there is Xenix (no DOS compatibility), which costs only seven times as much and takes up 6-8 megabytes of disk storage space vs. the 160K for Concurrent DOS.

"I am also a Unix advocate and have programmed under that OS for several years now. At home, however, I use Digital Research's Concurrent PC-DOS, Version 4.1. I find it to be a very good operating system environment with nearly all the profes-

sional systems design concepts described as essential for modern computing by such books as *Operating System Elements—A User Perspective* by Peter Calingaert (Prentice-Hall). Among the state-of-the-art features pointed out in this book that Concurrent PC-DOS provides and MS-DOS does not are:

- the concept of programs as finite-state processes within the computer
- dynamic memory management in the multitasking/multiuser environment
- systems support for QUEUEING structures (that is, interprocess message queues, priority queues, mutual exclusion queues, and so on)
- support for multilevel PRIORITY assignment to allow efficient scheduling of multiple concurrent processes
- inclusion of a directory hashing algorithm (which drastically increases system throughput)
- logical file and record locking
- multilevel passwording of files controlling read, write, and delete privileges
- time and date stamping of files (for creation, last access, and last modification)
- system support for processes to issue logical *wait* or *resume* interrupts—another efficient method used in interprocess communications
- real-time multitasking and multiuser support (note that Microsoft Windows uses a nonpreemptive multitasking algorithm, which means it cannot be used for applications requiring real-time response)
- logical data storage elements
- the ability to spawn multiple subprocesses from any single process (invaluable for any application that must monitor multiple port I/O such as in communications or instrumentation control)

"I agree and disagree with your statement that DRI will always be playing catch-up to Microsoft's MS-DOS. I believe that if you consider the features that DRI's PC-DOS-compatible OS family (which by the way runs on the 8088, 8086, 80286, and 68000) already provides, and which Microsoft is only now beginning to address for its MS-DOS, you might take a different attitude as to which operating system really provides the professional pro-

Make your PC or AT into a COMMUNICATING WORKSTATION for only $85

Use *ZAP*, the Communications System for Technical Users
COMPLETE Communications for PROGRAMMING and ENGINEERING

EMULATION of graphics and smart terminals is combined with the ability to TRANSFER files reliably, CAPTURE interactive sessions, and transmit MESSAGES while also being able to swap between your mini or mainframe session and your PC application. SUSPEND a line to run a PC application. Reconfigure features to fit the communications parameters and keyboard requirements of the host computer software. Complete technical documentation helps you understand and fit ZAP to your style.

HIGHLIGHTS OF *ZAP*:

- Emulate TEKtronix 4010/14 and DEC VT 100, 102, 52 including variable rows and columns, windows, full graphics, and smooth scrolling.
- Reliable ***file transfer*** to/from any mainframes and PCs including KERMIT and XMODEM protocols plus you get a full copy of KERMIT. Transfer ***speeds*** ranging from 50 to 38,400 BAUD. Session control include ***printer dumps*** and ***save to disk***.
- **MACRO and Installation files** ("scripts") controllable by you.
- EMACS, EDT and VI "Script" files are included. ZAP is also used with other popular software including graphics products like DISSPLA and SAS/GRAPH.
- **CONFIGURABLE** to communications, terminal features on the "other end"; 1, 2 stop bits; 5, 6, 7 or 8 data bits; parity of odd, even, none, mark and space; remap all keys including the numeric pad and standard keyboard, set any "virtual" screen size.
- Full **PC/MSDOS** access to run any command or program that will fit in your systems memory. ZAP takes less than 64K.
- 9 Comm ports are supported by ZAP. Plus full color in text and graphics make use of the IBM color, EGA cards, or Hercules Monochrome.

Full refund if not satisfied during first 30 days.

ONLY $85

Solution Systems

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle **no. 148** on reader service card.

The C Programmer's Assistant

C TOOLSET™

UNIX-like Utilities for Managing C Source Code

No C Programmer should be without their assistant – C ToolSet from Solution Systems. The package consists of several utilities designed to help make C programming tasks easier.

C ToolSet (formerly C Helper) includes:

DIFF – Compares text files on a line-by-line basis or use CMP for byte-by-byte – indispensable for showing changes among versions of a program under development. So "intelligent" it stays in synch even when you add 100 lines.

GREP – Regular expression searches – ideal for finding a procedural call or a variable definition amid a large number of header and source files.

FCHART – Traces the flow of control between the large modules of a program.

PP (C Beautifier) – Formats C program files so they are easier to read.

XREF (CCREF) – Cross references variables from a program.

Available For MS-DOS. – $95

Full refund if not satisfied during first 30 days.

ONLY $95
Source Code Included

Solution Systems™

335 Washington St.
Norwell, MA 02062
617-659-1571
800-821-2492

Circle **no. 152** on reader service card.

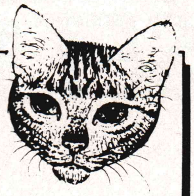
Try It. Then Buy It. *PC-Write.*™

A fast, full-featured word processing package for the unbelievable price of $10. Complete. You get a manual on disk, mail merge, split screen, keyboard macros, on-screen formatting, full printer support, and more.

Try *PC-Write* for $10.
Then register for $75 to get:
- latest diskette
- printed manual
- two updates
- phone support
- newsletter

Registration supports our "shareware" concept that keeps our prices low, and allows our development of *PC-Write* enhancements.

Shareware means you can get *PC-Write* from a friend or user group to try, and give away copies yourself. Then register if you like it. No risk!



Dr Dobb's Journal
July 1986

NOW
Version 2.6

The best package for both fast program text editing and nice looking documentation. **Version 2.7** with spell check coming soon, new price $89.

Order *PC-Write* Today.
Satisfaction Guaranteed.

Quicksoft™
(206) 282-0452
219 First N. #224L
Seattle, WA 98109

VISA MasterCard

Circle **no. 292** on reader service card.

16-BIT
(*continued from page 109*)

grammer with the more state-of-the-art programming environment!

"Yes, DRI is playing catch-up but only in an attempt to maintain file and programming compatibility with MS-DOS as Microsoft attempts to implement all of the aforementioned features. DRI's next release of the PC/XT version of this OS in two to three months and which is already in beta testing should provide the last vestiges of compatibility with MS-DOS, Version 2.1, including:

- full path support
- installable device drivers (the current version allows only hard-disk device drivers to be installed dynamically)
- DOS environment variable support via MS-DOS *set* command

"An extension to the version of the OS currently in beta testing, scheduled for summer 1986, is supposed to provide full MS-DOS, Version 3.1, compatibility!

"After all, if DRI has learned any lesson from the past four years, it has been that the public has been sold on the MS-DOS environment, and DRI must provide that functionality in its systems software. This only indicates to me that DRI is responding to the marketplace and if anything is looking backward to Microsoft's limited OS to see how it can be supported.

"Many a systems engineer will support the theory that a good operating system must be designed from the beginning to incorporate the structures necessary for real-time multiuser/multitasking. Attempting to do otherwise invites development of a most haphazard and kludged design in the attempt to maintain compatibility with whatever the current operating system provides. I think this problem with MS-DOS will become more evident as Microsoft attempts to introduce multitasking and multiuser support to its OS. I think it is relevant and most significant to notice that Microsoft did not provide its first attempt at multitasking through its operating system but through an externally run program—Windows.

"Although it is true that DRI has made several heroic blunders in the past, I do not believe they should be held as some sort of crucifix for it to bear. Everyone makes mistakes, and I do believe that DRI has and still is paying for its! I also believe, however, that it is the duty of columnists such as yourself to keep an open mind and to provide information that benefits the computing public at large and not to blindly assume such things as MS-DOS' a priori superiority of design.

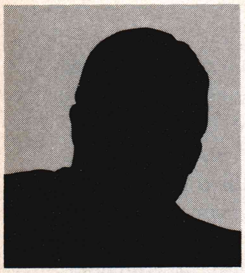
"You must remember that 90 percent of all businesses in the United States are small businesses that only have need for two to five terminal users. LANs are great and very cost-effective for six or more users, but for small businesses, a multiuser operating system that allows the attachment of $400–$500 dumb terminals to a single PC/XT/AT is a much more realistic and cost-effective approach to increased office productivity. The argument can be made for the use of multiple, cheap, clone PCs in such a situation, but how many businessmen would want multiple copies of their accounts-receivable file or inventory file on several different machines? I believe that multiuser operating systems such as Digital Research's Concurrent DOS definitely have their place in the world. They may not necessarily fit every situation or need, but then neither does the more simplistic environment provided by the current versions of MS-DOS."

I don't agree with all of Mr. Mullan's arguments, but for once I am going to pass up my prerogative as the columnist to have the last word on the subject. Further comments on the alleged need for multitasking, multiuser operating systems in personal computers are solicited from *DDJ* readers!

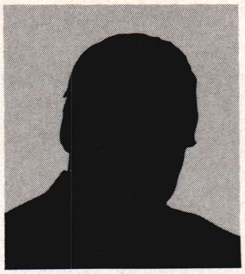
DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 9**.

"Multiple windows."

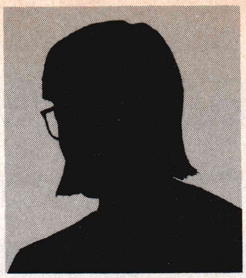


"Emacs runs on MS-DOS, VMS and UNIX. It's my company's standard editor."

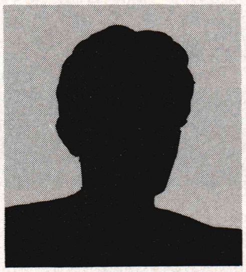


"I can run a shell or DCL command session even while editing."

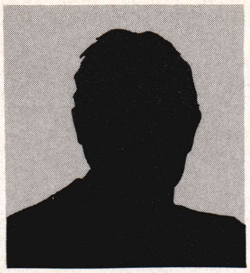
"A hundred times better than the UNIX vi editor."



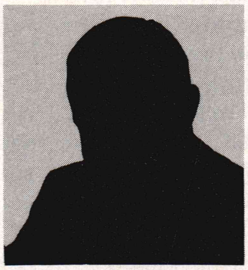
"Programming modes to assist me in writing C, Pascal, ADA and Lisp."



"UniPress really supports Emacs on Berkeley and AT&T UNIX, and is constantly developing Emacs."

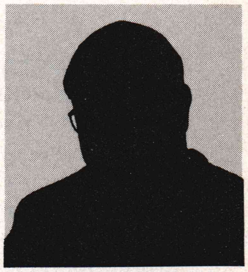


"I can associate any command to any keystroke sequence."



"Emacs automates the code-compile-edit-debug process."

"Electronic mail is only one of the many included packages I use."



"I can create my own customized editor commands with the MLisp extension language."

No Two Users Can Agree On Why UNIPRESS EMACS™ Is The Only Choice.

That's right. No two users can agree on why they've chosen UniPress Emacs. UniPress Emacs provides so much for creating and modifying programs, documents and other text files. No other program has such powerful facilities. And we offer Emacs for UNIX,™ Xenix,™ Ultrix,™ MS-DOS™ and VMS™ systems.

UniPress Emacs gives you all the basic text editing commands, and goes much further. It allows multiple windows to display several files or portions of the same file simultaneously, a program's output, a shell/DCL window, a help window, an error listing, an electronic mail message, or anything else!

UniPress Emacs simplifies programming with its C, Pascal, ADA and Lisp modes. It checks for balanced parentheses and braces, and indents and reformats code as needed. "C mode" produces templates of control flow in user-definable styles. After compiling, Emacs collects any compiler error output in a window. Emacs then analyzes this error output and automatically displays the erroneous source lines, one-by-one.

In addition, Emacs offers great extensibility through macros and the built-in compiled MLisp™ programming language. The MLisp language provides looping, conditional testing and other flow control statements, and a rich set of logic, arithmetic and string operations which can be combined.

Circle **no. 77** on reader service card.

Trademarks of: UniPress Emacs & MLisp, UniPress Software, Inc.; UNIX, AT&T Bell Laboratories; VMS, & Ultrix, Digital Equipment Corp.; MS-DOS; Xenix, Microsoft.

If you're still not convinced about UniPress Emacs, call us at **800-222-0550** (Outside NJ) to find out about our **dial-in demo** machine or send in the coupon to get more product information. We'll send you our **free catalog** with information about Emacs and our other software products.

UniPress Software, Inc., 2025 Lincoln Highway, Edison, NJ 08817. Telephone: 201-985-8000. Order Desk: 800- 222-0550 (Outside NJ). Telex: 709418

European Distributor: Modulator SA, Switzerland. Telephone: 41 31 59 22 22. Telex: 911859

Japanese Distributor: D.I.T. Inc., Minato-Ku, Tokyo. Telephone: 03 586-6580

UniPress Software

Your Leading Source for UNIX Software.

Put Yourself in the Picture! Discover Why UniPress Emacs is the Only Choice in Text Editors!

Name: ____________________

Company: ____________________

Address: ____________________

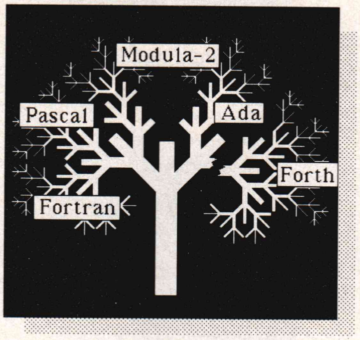
City: __________ State ____ Zip ____

Phone: ____________________

DD-786

Circle **no. 77** on reader service card.

Forth: Philosophy, Standards, and Practical Advice



Programming is control; languages that most directly give the programmer control touch most deeply the programming impulse. That's the appeal of assembly language: *mano a mano* with the machine's bare metal. Those who prefer high-level languages find in Forth an edge-of-the-envelope programming environment. What's it like out there?

Forth puts the programmer into intimate contact with the insides of the language, unlike other high-level languages whose compilers are black boxes. The Forth compiler consists of instructions in Forth itself—the compiler's *right there*, where the programmer can reach out and touch it, not locked away on the other side of some impenetrable barrier. Reach out, touch it, and put a little spin on it.

Programming in Forth consists of adding new commands to the language. The programmer-created commands climb latticelike to the solution. The command at the top is the program; when executed, it answers the question posed by the task. This topmost command is defined in terms of commands lower in the hierarchy, and those commands are defined in terms of commands lower still. (Though recursion in Forth is possible, it is not used as often as it is in, say, LISP.) The program's foundation is a small set of primitives written in the language of the computer on which the program will run.

The program may also contain a few assembly-language commands

by Michael Ham

added by the programmer at speed-critical parts of the application. Most of the program, however, will consist of commands written in Forth:

© *by Michael Ham. All rights reserved*

some constitute the Forth nucleus, some are the programmer's own general-purpose tool words, and the remainder are specific to the problem at hand. Some of these programmer-created commands extend the compiler. But first let's look at the programmer's tools.

Over time, Forth programmers create or collect a set of tiny tools that they graft onto the Forth they use. (Every Forth implementation provides a way for its user to customize it through the permanent addition of new commands.) Tiny tools percolate through the community of Forth programmers and in time become assimilated as regular Forth words. *NIP* and *TUCK* are close to becoming a part of generic Forth. A few tiny tools are shown in Listing One, page 104; do you know their authors? Before their origins are forgotten, we should acknowledge their creators and refiners. Send in your own favorite tiny tools (with attribution if you can), and I'll publish them for general consumption to expedite the diffusion.

In addition to these tiny tools, programmers build larger tools as well. These are generally shaped by the kind of applications the programmer writes. Later in this column, I describe (as a challenge to you) a number-input tool I have developed. I would be interested in seeing your solution. I'll include the best solution I get (or, by default, my own effort) in a future column.

Extensions to the compiler (which at first I avoided altogether) are the source of some of Forth's magic. Extensions are of two types: the *CREATE . . . DOES>* defining commands normally used to add new data classes and the *IMMEDIATE* words that can be used to make new control structures. Let's examine the *CREATE . . . DOES>* commands now and save the *IMMEDIATE* guys for a future column. With a defining command you can create a family of structures in which all members obey the rule laid down in *DOES>*.

Suppose, for example, you will be using a variety of arrays in some application. Most Forths don't provide an array-defining word: an array-creating command is easy to write, and different situations demand different array specialties. Some programmers might want the array to do a range check on indices; others (such as myself) prefer to do any necessary edits before the indices are passed to the array. When some internal program routine generates the indices within definite bounds, range checking would sacrifice speed unnecessarily. I discuss a series of variations on an array-defining command later in this column.

Because Forth doesn't have a "hands-off" compiler whose internals are secret, Forth application programmers are led within the language. They work outward toward their application, but at the same time they work inward to tinker with the language itself. You don't even have to scratch a Forth application programmer to find the systems programmer within: the systems programmer is right there, working alongside the application programmer. In the array-defining commands, for example, you work toward adding a kind of type declaration to the language, a function already embedded within the compilers of many languages: in those languages, however, you usually don't have a chance to make it work in the way you want.

Every Forth programmer spends some time working on the language

itself as the application. The systems enhancements become part of her or his toolkit for future work, and Forth grows vinelike to enclose the programming problems most often encountered.

Forth is by design an open-architecture language. An accident of history plunged Forth programmers deep into that architecture from the beginning. The first Forth products, sold by FORTH Inc., were beyond the financial reach of hobbyists. So a band of programmers, the original Forth Interest Group (FIG), developed public-domain Forth systems to promulgate the language to hobbyists and hackers. The fig-Forth systems were distributed in the form of source-code listings, so early Forth users were willy-nilly systems programmers as they typed in the code and tuned it to their needs. Many commercial Forth products were built upon and grew out of these early fig-Forth listings. Control of the machine was sublimated into control of the language.

Forth Standards

Because Forth so readily grows in every direction, standards were needed to define a common core. Three standards have come to be, the later supplanting the earlier. The first was the FIG standard, a de facto standard created by the popularity and widespread distribution of those early FIG listings. This was followed by the 79 Standard, the first attempt at a formal and deliberate standard. The 79 Standard benefited from hindsight: It contained what the original fig-Forth would have included if its creators had had more Forth experience instead of learning while doing. The latest standard is the 83 Standard—an effort to polish, refine, and extend the 79 Standard.

In most of the computer industry's standards efforts, intense conflicts arise between unassailable logic (my position) and pointless pig-headed resistance (your position). This dynamic could also be detected in the development of the 83 Standard. When the 83 Standard was announced, the Forth community's delight at discovering refinements (read: changes) from the 79 Standard was muted. The spirit of innovation had acquired a conservative cast.

Learn and Use AI Technology In Your First Evening With PROLOG-86

NOW ONLY $95

A complete *Prolog Interpreter, Tutorial,* and set of *Sample Programs:*

☐ **Modify and write Expert Systems.** Use the simple "Guess the animal" example on the Tutorial or use the sophisticated system for Section 318 of the US Tax Code written by one of the PROLOG-86 authors and published in the March, 1985 issue of Dr. Dobb's Journal.

☐ **Write Symbolic Math or Abstract Problem Solving Applications** This is a complete Prolog program to convert from Farenheit to Centigrade: f_to_c(C,F):- C is(F-32) *5/9. Planning programs and games are included to help you learn.

☐ **Understand Natural Language** Use the sample program that produces a dBase DISPLAY command as output.

☐ **BECOME FAMILIAR WITH PROLOG IN ONE EVENING.**

Programming experience is not required, but a logical mind is.

Prolog-86 Plus for $250 adds: Windowing, 8087, 640K memory access, random access files, strings support and definate clause grammar.

RECENT IMPROVEMENTS: Floating point support, MSDOS commands, on-line help, load Editor.

AVAILABILITY: All MSDOS, PCDOS systems.

Full refund if not satisfied during first 30 days.

ONLY $95

Solution Systems™

335-R Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle **no. 153** on reader service card.

Brand New From Peter Norton A PROGRAMMER'S EDITOR

that's *lightning fast* with the *hot* features programmers need

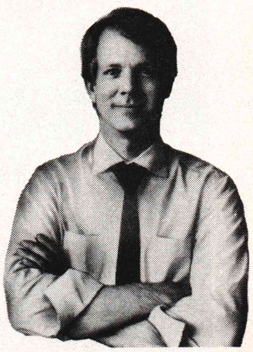
only $50

THE NORTON EDITOR™

Direct from the man who gave you *The Norton Utilities, Inside the IBM PC,* and the *Peter Norton Programmer's Guide.*

"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities.* You can ***program your way to glory*** with *The Norton Editor.*"

Peter Norton



Easily customized, and saved
Split-screen editing
A wonderful condensed/outline display
Great for assembler, Pascal *and* C

Peter Norton Computing, Inc., 2210 Wilshire Boulevard, Santa Monica, CA 90403, 213-453-2361. Visa, Mastercard and phone orders welcome.

The Norton Editor™ is a trademark of Peter Norton Computing, Inc. © 1986 Peter Norton Computing.

Circle **no. 243** on reader service card.

STRUCTURED PROGRAMMING
(continued from page 113)

Those moving up to 83 Standard Forths should take note of Ray Duncan's article "Converting FIG-Forth to Forth 83" in the May 1984 *DDJ*, Robert Berkey's two articles entitled "79-Standard to 83-Standard" in *Forth Dimensions* (vol. 6, nos. 3 and 4), and Kevin McCabe's review of the 83 Standard in the August 1984 issue of *Byte*.

The 83 Standard defines the behavior of a core of Forth words. Several different Forths adhere to the 83 Standard; their standard words work in the prescribed manner, though they may differ in speed of execution because of the differences in how they were implemented. The extensions (such as the file-support system) to these Forths are not governed by the standard and thus may differ considerably, and a particular Forth may be tuned to a specific machine—for example, it might support function keys and a speaker, both of which go beyond issues addressed by the standard.

Several implementations of 83 Standard Forth are available. Laboratory Microsystems and Micromotion have 83 Standard Forths for a variety of computers. Harvard Softworks has an overlay file that makes its Forth meet the 83 Standard. F83, a Forth written by Henry Laxen and Michael Perry, is a public-domain version of an 83 Standard Forth.

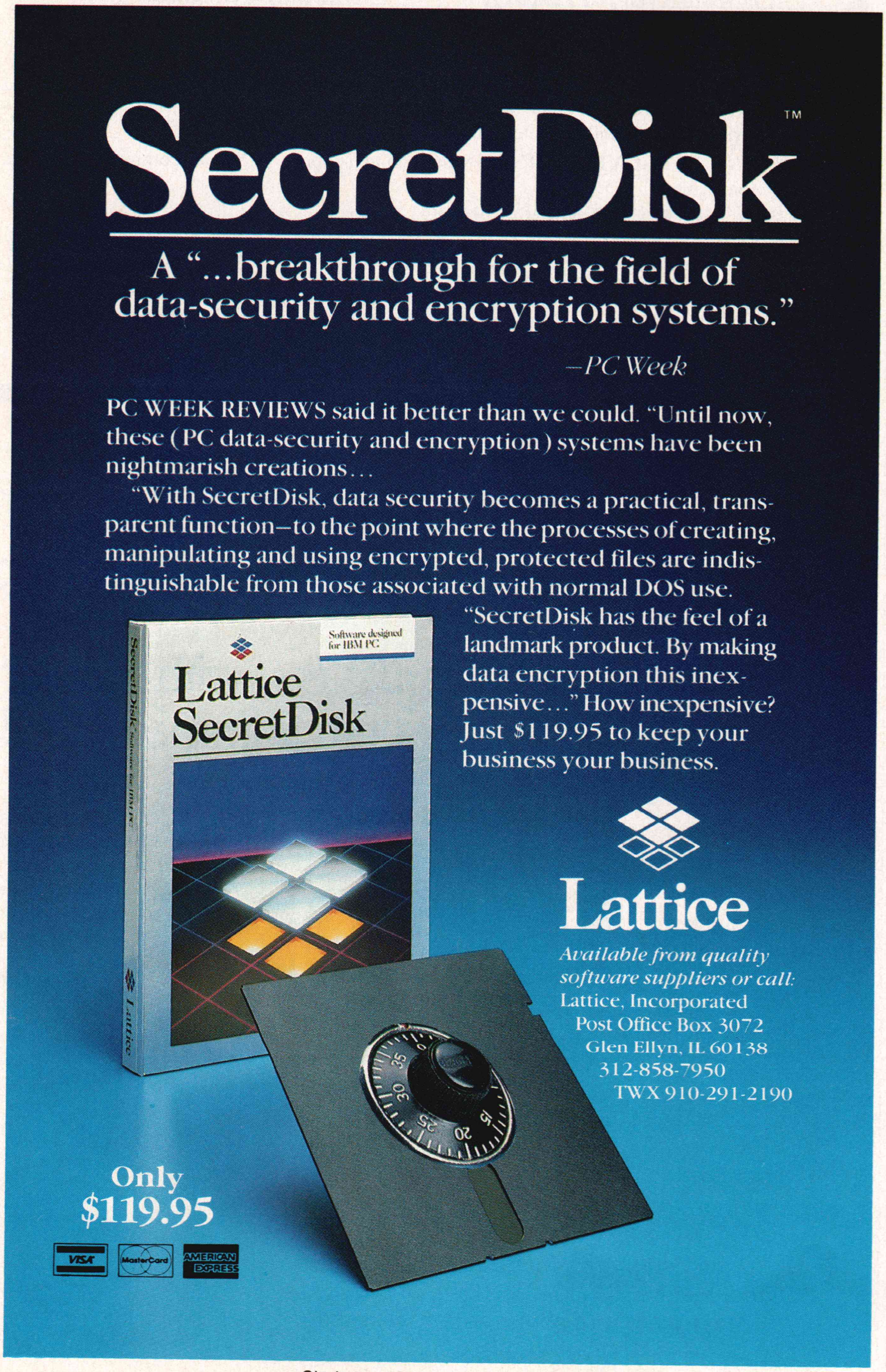
Because the 83 Standard specifies only a 16-bit implementation, 32-bit Forths (which can address more than 64K of instruction space) are by definition nonstandard. Some 32-bit Forths, however, (such as Palo Alto Shipping Company's Forth for the Macintosh and LMI's Forth+ products for the 8088/8086 and the 68000) strive to be standard in all other respects.

Although the 83 Standard will probably stand as the latest effort for quite a while, it fails to address some important topics: floating point (a de facto standard seems to be emerging), graphics, and a standard implementation for Forths that inhabit memory beyond 64K. These issues are necessarily being addressed by Forth vendors, and a consensus may in time emerge and be recognized in a later standard.

Where Is It Used, and Who Uses It?

Forth is well known as a language for data acquisition and machine and process control and is often the high-level language hiding inside the ROM of an intelligent machine. But Forth is found laboring in other vineyards as well. Business applications, for example, are not commonly thought of as Forth territory, but Forth was in business from its commercial birth. When Forth emerged from its womb of astronomical telescope control and was delivered to FORTH Inc.'s first customer, it was for a business database system in a custom application. How many of you are writing Forth code to address applications that might be thought of as food for COBOL?

For that matter, what is the range of application areas addressed by Forth? Many software products don't reveal their lingual origins unless the language in question is riding high or deemed especially appropriate for the given application area. Do you



Circle **no. 101** on reader service card.

know accounting packages that are unexpectedly Forth at heart? Databases? Games? Instructional programs? I would like to hear about intriguing uses of Forth. Send me information that I can print, and let's see what an honor roll of Forth programs looks like.

Forth programmers are an interesting lot. Compared to programmers of most other languages, relatively few Forth programmers have degrees in computer science. Many (most?) are self-taught programmers: they had a project that involved some programming and found Forth a good tool—easy to learn, quickly productive, and well adapted to an experimental approach. These characteristics result from Forth's evolution: a working language shaped in the field by a programmer who had to fit big solutions into small computers in little time.

Forth programmers include a high proportion of hardware-oriented people who enjoy working at the boundary between software and hardware, using knowledge gained by experimentation and experience. Forth programmers abhor protective protocols and demand the ability to grab the guts of the hardware and software. They are an innovative lot, generally impatient with theory in their eagerness to get the job done. They are willing to share their idea but are sometimes reluctant to change their own approach. This leads to the reinvention as well as the refinement of techniques.

Where Can You Talk with Them?

Forth people are scattered across the country. Many are members of the Forth Interest Group [P.O. Box 8231, San Jose, CA 95155; (408) 277-0668]. FIG publishes a journal and also has a 300-baud bulletin board for discussions (no files). It's at (415) 962-8653—press two carriage returns when you connect. FIG's local chapters across the country provide opportunities for face-to-face contact.

DDJ includes Forth offerings in its own CompuServe forum (GO DDJ at the ! prompt). Creative Solutions, a Forth vendor, also hosts a Forth discussion group on CompuServe (GO FORTH). Two exclusively Forth bulletin boards are the East Coast Forth BBS [(703) 442-8695] and the West Coast LMI BBS [(213) 306-3530]. Both operate at 300/1200/2400 bps, are available around the clock, and include both discussion and files of source code.

A Look at Arrays

Array-defining words are the five-finger exercises of defining words. It's easy to extend the compiler to build arrays that match your taste and needs exactly. Listings Two through Eight trace one path of development for an array-defining word.

Listing Two, page 104, shows a first attempt. Though this is certainly good enough for workaday use, it has a couple of drawbacks: it defines arrays only for single-precision numbers, and the use of *ARRAY* for the name, though an obvious choice, results in awkward and un-English phrasing in the source code.

You will note that the stack comment following *DOES>* contains one address in angle brackets. This is the address that *DOES>* places on the stack at execution time. I look at the stack comments as I write the definition, and if I slip and forget that this address is present, I write bugs. I put the address in brackets to indicate that it is supplied by the word itself.

Listing Three, page 104, shows a more flexible array-defining word: It can create an array of bytes, cells, or double-precision numbers. The name *FOR* is used instead of *ARRAY* to make the code read more naturally. Picking good names for Forth words is a difficult art; you can be led astray by an impulse to name words according to their internal implementation rather than with an eye to their use.

At creation time, *FOR* stores the array type (the number of bytes an entry occupies) into the first byte of the array area and initializes the rest of the array to 0s. The index is presented at execution time, and the type is retrieved (with *COUNT*). The product of the type and the index then gives the offset to the entry, and this offset (possibly 0) is added to the address of the first entry.

This definition, however, makes the programmer pick the correct storage or retrieval word. Every decision the programmer makes represents an opportunity to decide incorrectly. Moreover, if I happen to use the wrong storage or retrieval word, it is hard for me to spot the error. If the right kind of word is in the right place, it looks—well—right. If I use *@* with a byte or double-precision array (when I should have used *C@* or *2@*), the result is wrong, but I have a devil of a time seeing the error, even when I am looking at it.

Listing Four, page 104, shows one solution: give that responsibility to the array word. The flags *PUT* and *GET* determine the direction in which the datum will move. When the array word is executed, the *DOES>* clause of its parent *FOR* uses the type number (a copy of which it momentarily stashes on the return stack while it does other work) to dip into either *STORES* or *FETCHES* to retrieve the correct operation to perform. (The command *]* turns on the compiler, and so the stores and fetches following are not immediately executed; rather, their (2-byte) addresses are stored into the dictionary. The *[* turns the compiler off again.)

I often use bit arrays to save room. Listing Five, page 104, shows my collection of bit tools; these appeared in a slightly different form in an article in *Computer Language*. The prefix + for "turn on" and − for "turn off" follow naming conventions suggested by Kim Harris (see later). I use ~ as a prefix meaning toggle. *@BIT* uses the word *FLAG* to force nonzeros to the 83 Standard Boolean value for true (−1) so that the fetched value will work appropriately with logical operators such as *AND* and *OR*. You should note that the *NOT* in the definition of *−BIT* is the 83 Standard *NOT*, which operates bitwise (as do the other logical operators *AND*, *OR*, and *XOR*). The 79 Standard *NOT* was merely a synonym for *0=*. If you have not yet moved to an 83 Standard Forth, you should replace *NOT* with *−1 XOR*.

The *FOR* in Listing Six, page 104, can also create bit arrays. For arrays of bytes, cells, or doubles, this *FOR* works exactly like the *FOR* in Listing Four does. The range of values that can be stored in a bit is limited, so I embed the *PUT* function in *SET*, *ZAP*, and *FLIP* to make the phrases read better in the bit context.

At one time I would have stopped here. But Kim Harris has pointed out that whenever you create a new data structure, you should also create a word to display its contents. These

(continued from page 115)

"inspection" words inevitably repay their development cost as soon as you begin to use the new structures. (Think of trying to use the stack without *.S* to let you look at what's there.)

So I go one step further. The display word needs to know how many elements to display, so a new *FOR* is shown in Listing Seven, page 105. This *FOR* stores the number of slots as part of the array information. I then need a word to move past the cell holding the array size to the byte where the type is stored. Rather than put this step in the definition as a (subsequently) mysterious *2+*, I factor it out for separate definition as *>TYPE*. If I move to a 32-bit architecture, I'll know to modify this word accordingly. *>DATA* is defined similarly.

My first impulse was to have *SPILL* just be a constant with a negative value. *FOR*'s *DOES>* clause would then be modified to first check the sign of the index. If the index was positive, *DOES>* would do its usual work of storage and retrieval; if the index was negative, *DOES>* would know that what was wanted was a display of the array contents and it would include the code for that.

I rejected this approach, even though *FOR*'s definition could still be made readable by factoring out some of the subfunctions. The display function is only for development, but *FOR* is a production word and so it should not include development tasks. By defining the display word separately, I don't have to include it in the production version of the program.

Listing Eight, page 105, shows the development of *SPILL*, which expects to be followed by the name of an array. *SPILL* displays only five double-precision numbers per line because of their potential length; otherwise ten array entries are shown per line. *}LINE* starts a new line when appropriate. My naming convention for words that execute conditionally is to use *}* as a prefix. This doesn't match the example of *?DUP*, but I prefer to restrict my use of the prefix *?* to words that expect a flag. You can read *}* as "maybe."

In my first iteration of developing the bit display, bits were shown in the "natural" way, as 0s and 1s. [Because *@BIT* converts bits to Boolean flags (0 and −1), I used *NEGATE* to produce more conventional bit values (0 and 1) for the display.] On testing the word, however, I found that I couldn't see the 1s for the 0s. So I revised the display to show "off" bits as hyphens. The 1s of the "on" bits then stand out nicely.

I assumed that an array will have fewer than 10,000 elements, and so the line label is set to have at most four digits. In fact, the display word, like the array word, reflects any number of assumptions about how the data should be presented. These reflect my taste and requirements. (Two examples in addition to those mentioned above: even though the final *FOR* knows the number of slots in the array, I still prefer that the array word not perform a range check on the index; and single-precision numbers are displayed with *.* instead of *U.* because the arrays I use are more likely to contain negative numbers than addresses.) It is the strength of Forth that you can tailor these tools to suit yourself.

Naming of Parts

I have learned the hard way that *JOAN* and *JOHN* would be poor names for program words. Not only do they fail to tell the reader what is going on but they also are spelled the same except for the difference of a single letter and (worse) they are the same kind of word. You might type "H" when "A" is intended. The error will be accepted because *JOHN* is a valid word. The program will even work after a fashion because *JOHN* and *JOAN* fulfill similar functions. And once again I would be trying to find a bug that consists of the right kind of word—but not the right word—in the right place. Whenever possible, I make sure that names differ by more than a single letter.

Another poor name choice that I considered briefly was *TO* for *PUT*. The problem with *TO* is that it is a homonym for *2*. Homonyms are an annoyance when you try to talk about the code.

If you ever use hex, it's also a bad idea to use names that could be numbers. This problem can be alleviated by making it an absolute rule to write hex numerals with a leading 0.

A pattern of naming Forth words has developed slowly. Kim Harris has compiled a reasonably large set of naming conventions that seem to be generally accepted. These have been published as an appendix to Leo Brodie's *Thinking Forth* (Englewood Cliffs, N.J.: Prentice-Hall, 1984) and as papers in the *1984 Rochester Forth Applications Conference Proceedings* and the *1983 FORML Conference Proceedings.*

A Number-Input Word: Challenge to Readers

Programmers often want users to enter numeric information. The challenge is to develop an easy-to-use command with a user-friendly face. I'll now discuss some suggested design specifications.

Each digit is displayed on the screen as it is entered. The display is "calculator" style, with digits appearing (and disappearing) at the rightmost end of the number. The routine inserts commas in the display as needed. The minus key operates as a toggle (which accommodates the usual entering the minus sign at the beginning but also permits it to be turned on or off after the number is underway). Backspace and Delete (and any other left-arrow key you might have on your computer) rub out the rightmost character (which might be a minus sign or a decimal point). The remaining characters in the field shift one place to the right to fill the gap. Entering "B" or "C" (uppercase or lowercase) erases everything that has been entered. Entering any illegal character or attempting to delete when the field is blank results in a beep. Thus every key either produces some alteration in the display or sounds a bell.

If the user enters a 0 to start, entering additional 0s immediately thereafter does not result in a repeating series of 0 (unless, of course, a decimal point was entered first). Other digits do repeat. As a courtesy to the user, the letter *l* (uppercase or lowercase) is accepted as the number *1* and the letter *o* as the number *0*.

The programmer specifies whether a minus sign is allowed; if it is not, pressing the minus key produces a beep but no entry. Similarly, if the programmer indicates no decimal places, the decimal point is beeped as

invalid input.

The format of the commands the programmer uses to manipulate the routine in his or her program is as follows:

1. n PLACES—*n* is the number of places to the right of the decimal point. The value presumably is stored in a variable. The number the user enters is collected as a double-precision integer, so the number of decimal places is a scaling factor. A decimal point is a legal character for the user to enter only if the number of places is greater than zero. The default is zero places. If the user presses Return after entering only the whole-number portion of a number with decimal places, the decimal point and trailing 0s are supplied by the routine.
2. NEG-OK ON—*NEG-OK* is a variable. The default value is false (off). Minus signs are accepted only if *NEG-OK* is on.
3. d n −1 DIGITS or n 0 DIGITS—The true/false flag (embedded in some mnemonic name) tells the routine whether it is being supplied with a number to begin with. If the flag is true, the (double-precision) number is displayed in the entry field (with commas, minus sign, and decimal point as appropriate) for the user to accept or alter as needed. If the flag is false, no number is supplied and the routine begins with a blank entry box.

The single-precision number *n* specifies the total number of digits the user may enter. This value, together with the number of places, determines the number of digits allowed to the left of the decimal point. The sequence *2 PLACES 5 NEW DIGITS*, for example, means that there will be at most three digits to the left of the decimal point and at most two digits to the right. (*NEW* here is assumed to be a constant equal to 0.)

DIGITS presents an inverse-video field at the current cursor location. The field allows room for minus (but only if *NEG-OK* is on), for commas (but only as many as can be entered given the number of digits allowed), and for a decimal point (but only if more than zero places have been specified). The field includes a blank inverse space before and after the spaces needed to hold the number, sign, commas, and decimal point of the number being entered.

When the user signifies the end of input by pressing the carriage return, the stack contains a double-precision number (the value of the entry) under a single-precision number (the number of digits the user actually entered). This allows the program to distinguish "no entry" from an entered 0.

Send me your own solution, preferably in an 83 Standard Forth. In a future column, I'll take a look at the results.

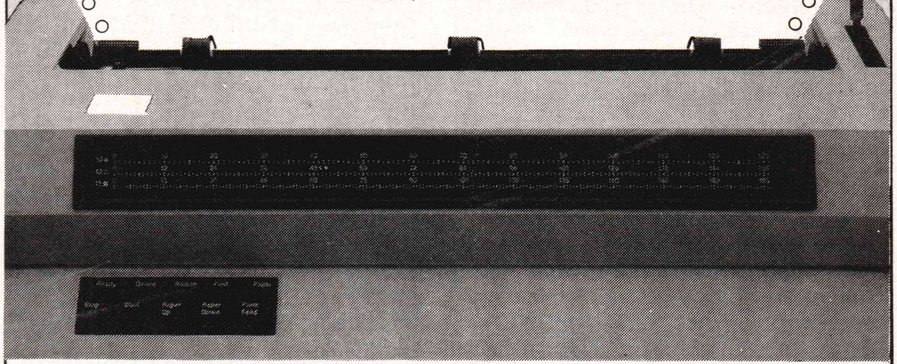
Operating Systems and Text and Block Files

Forth was originally its own operating system: It seized control of the entire computer and handled everything itself. This Forth operating system included a simple and effective way to access the disk directly using a block number. Each block was 1K, and source code was displayed on the screen in one-block chunks, so blocks were also called "screens."

As the micro world grew to include more applications and hard-

C Programmer Essentials

"Offers many capabilities for a reasonable price"
W. Hunt, PC Tech Journal
"I highly recommend the C UTILITY LIBRARY"
D. Deloria, The C Journal



C ESSENTIALS **$100**
200 functions: video, strings, keyboard, directories, files, time/date and more. Source code is 95% C. Comprehensive manual with plenty of examples. Demo programs on diskette. Upgrade to THE C UTILITY LIBRARY for $95.

THE C UTILITY LIBRARY **$185**
Thousands in use world wide. 300 functions for serious software developers. The C ESSENTIALS plus "pop-up" windows, business graphics, data entry, DOS command and program execution, polled async communications, sound and more.

ESSENTIAL GRAPHICS **$250**
Fast, powerful, and easy to use. Draw a pie or bar chart with one function. Animation (GET and PUT), filling (PAINT) and user definable patterns. IBM color, IBM EGA and Hercules supported (more soon). NO ROYALTIES. Save $50 when purchased with above libraries.

Compatible with Microsoft Ver. 3, Lattice, Aztec, Mark Williams, CI-C86, DeSmet, and Wizard C Compilers. IBM PC/XT/AT and true compatibles.

C Compiler Packages: Microsoft C - 319, Lattice or CI-C86 compilers -$329. Save $40 - $50 when purchasing compiler and library combinations. Specify C compiler and version number when ordering. Add $4 for UPS or $7 for UPS 2-day. NJ residents add 6% sales tax. Visa, MC, Checks, PO's.

ESI ESSENTIAL SOFTWARE, INC
P.O. Box 1003 Maplewood, NJ 07040 914/762-6605

Circle no. **138** on reader service card.

Put More UNIX™ in Your C.

Unitools $99
MAKE, DIFF and GREP
These versatile UNIX-style utilities put power at your fingertips. MAKE, a program administrative tool, is like having an assistant programmer at your side. DIFF compares files and shows you the differences between them. GREP can search one or many files looking for one pattern or a host of patterns.

"Z" $99
A Powerful "vi"-type Editor:
Similar to the Berkeley "vi" editor, "Z's" commands are flexible, terse, and powerful; macro functions give you unlimited range. Features include "undo," sophisticated search and replace functions, automatic indentation, C-tags, and much, much more.

PC-LINT $99
Error Checking Utility
A LINT-like utility that analyzes programs and uncovers bugs, quirks and inconsistencies. Detects subtle errors. Supports large and small memory models, has clear error messages and executes quickly. Has lots of options and features that you wouldn't expect at this low price.

SunScreen $99
Low-priced Screen Utility
This versatile graphics package easily creates and modifies formatted screens, validates fields, supports function keys, color and monochrome cards. With library source SunScreen is $199.

Compatible with all leading MS/PC-DOS C compilers.

SPECIAL OFFER: Unitools, "Z," PC-LINT and SunScreen All for only **$349**

To order or for information call:

TECWARE
1-800-TEC-WARE
(In NJ call 201-530-6307)

UNIX is a registered TM of Bell Laboratories. MANX AZTEC TM Manx Software Systems, Inc. PC LINT TM GIMPLE software. SunScreen TM SunTec. MS-DOS TM Microsoft.

Circle **no. 223** on reader service card.

STRUCTURED PROGRAMMING
(continued from page 117)

disk storage became more common, the omnipresence of the operating system became inescapable. Few users were willing to reboot between applications simply so that Forth could run its own show, and fewer still wanted to partition their hard disk to allow different disk formats to share the space. (Embedded applications do not have this problem, of course, and the Forth operating system is still often found in that environment as well as in totally dedicated systems, such as for process control and data acquisition.)

Forth now commonly runs under an operating system and makes use of the operating system's I/O services. Blocks are still used, but in this environment they reside within files.

The efficacy of the block files is a topic of perennial debate. Some programmers prefer text files so that they don't have to group the source code into 1K blocks. Others accept the modular discipline of the 1K chunks and as a bonus retain the the ability to incrementally compile a module under development. That is, they load and test a block until it works; write, load, and test the next block; and so on—instead of reloading the entire file with each change. Block files, with their separately loadable blocks, fit the interactive program development style that is Forth's special tactic.

One practice sometimes seen in block files—using block-number ranges to create subfiles within a file—seems worth discarding. Within a block file a programmer might use, for example, blocks 5–10 are for one module, 15–25 for another, 30–40 for a set of data pointers, and block 50 and beyond for the data. These block-number ranges are a hangover from the "file" systems typically used in native-mode Forths, when the only disk access was by block number.

The gaps in the block ranges are intended to simplify expansion and maintenance of the "subfile" system. Because Forth stores 1K per screen, though, this technique eats up too much storage room in operating systems such as MS-DOS or PC-DOS, where files cannot have gaps. Moreover, adding blocks to accommodate new functions will often throw a monkey wrench into the numbering scheme and invalidate the block numbers in the load block.

It is simpler and more efficient to exploit the strengths of the file system and use different files for separate submodules of source code. Forths running under an operating system will include a complement of file-handling words so that the program can open and close files as needed. Different files of Forth source code, whether block or text files, can be called during a load sequence with some word such as *INCLUDE*. The load block, instead of specifying block number ranges, can *INCLUDE* specific file names, and those files can expand or contract as the program is maintained and revised, with no effect on the load instructions.

Indeed, if a particular submodule (for example, the number-input routine described earlier) turns out to be generally useful, it is quite handy to have it as a file of its own to be *INCLUDE*d in as many programs as needed. Some Forths allow the developer to create small, relocatable binary overlays so that such modules can be called quickly and serve as a component of a Forth library of tools.

Factoring modules into separate files is an extension of the idea of factoring functions into separate words. A Forth programmer learns through experience when a routine deserves its own identity, whether as word or as file.

There's no denying, however, that blocks, whether in files or in native mode, consume a lot of disk space. Most of this is because of their puffiness: a block occupies 1K of disk space even though a considerable part of that 1K may be blanks. That's why Forth programmers dearly love the archiving programs that squeeze out the blanks when the file is stored in archive format. My archived files are typically 20 percent of their original size. (This, of course, includes compression beyond merely squeezing down the blanks.)

The program I use is a shareware program called ARC. It is under continuing development by System Enhancement Associates (21 New Street, Wayne, NJ 07470). You can buy it from the firm for $50, but it prefers

that you obtain the program through the normal shareware distribution channels. Version 5.1 was released in January, but new versions appear with disconcerting frequency. A version can read and decompress archive files created by any lower-numbered version but not vice versa.

ARC builds archive files that contain as many compressed files as you want. You can add files to an archive file and extract them quickly whenever you need them; it is also easy to update an archived file with a later version. The use of ARC makes the bulkiness of the block file much less of an issue.

Background

I plan to be a regular denizen of these pages, so you might want some idea of my programming background and Forth experience. I started programming on an IBM 1401, and the language was Autocoder. From there I moved on to FORTRAN, dipped into BASIC, glanced at APL and Pascal, and at last discovered Forth through an article in *DDJ* several years ago.

I got a copy of Miller Microcomputer Services' MMSForth and found Forth irresistible. Because I wanted to share and sell my programs, I moved to Forth Technology's Forth/Level 2, a spin-off of FORTH Inc.'s polyFORTH. Forth/Level 2 required no license fees or royalties, and its *TURNKEY* word was an easy way to produce bootable programs.

Finally, however, I had to recognize that the world of business applications in which I worked was increasingly dominated by PC-DOS/MS-DOS. Native-mode Forths did not fit that environment comfortably, particularly as hard disks became more common. I moved to Laboratory Microsystems' PC/Forth because it worked well with DOS and because it was one of the few vendor-supported, 83 Standard Forths then available. It also required no license fees or royalties for turnkeyed programs.

I did not really consider a public-domain Forth. Because I write programs for a living, I want vendor support. I don't want to be the one who has to write every extension package and constantly track new technology and adjust my system to fit. I feel that my time is better spent on billable projects, and I am willing to pay the minor upgrade charges to have the vendor keep the system tuned to new machines and new versions of the operating system.

Currently I am completing a reasonably large Forth application (about a thousand screens), a software package that will be published RSN. I am also an outside contractor for Laboratory Microsystems, assisting with technical support and documentation—a natural progression from my own extensive use of the company's technical support.

From my experience, I know the above-mentioned Forths better than I do others, but I am sure that my readers will expand my horizons. You are welcome to write to me on the DDJ Forum on CompuServe or care of the magazine. I look forward to hearing from you.

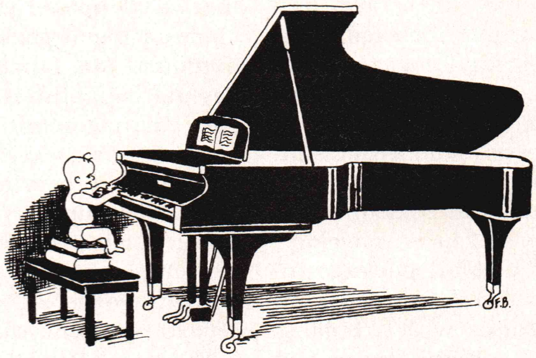
DDJ

(Listings begin on page 104.)

Vote for your favorite feature/article.
Circle Reader Service **No. 10.**

C–terp

The C Interpreter You Won't Outgrow



C–terp will grow with you as you progress from novice through professional to guru. Unbelievable, but true, the easiest-to-use C interpreter will provide you with the most advanced programming features for upward growth. Our exclusive **object module support** enables you to add libraries (like HALO, PANEL, Windows for C, etc., or your own homebrew libraries) to C–terp as you add them to your computing repertoire. Use C–terp as a microscope on your libraries! Flip a bit and allow our **software paging** (NEW) to handle those big jobs! There are no fixed-size tables to overflow, and C–terp can be configured for different screens and screen adapters (NEW). With multiple modules and **full K&R support,** we offer a dream C environment.

- Our new improved **configurable editor** competes with anything going.
- Speed -- Linking and semi-compilation are breathtakingly fast.
- Convenience -- Errors direct you back to the editor with the cursor set to the trouble spot.
- Symbolic Debugging -- Set breakpoints, single-step, and directly execute C expressions.
- Compatibility guaranteed – batch file to link in your compiler's entire library. Supported compilers include: **Computer Innovations C86, Lattice C, Microsoft C 3.0, Mark Williams C86,** and **Aztec C.**
- Many more features including batch mode and 8087 support.

What Our Users/ Reviewers Are Saying

"... easy to use, powerful, and a timesaver."

"... we absolutely LOVE C–terp."

"... has restored my faith in interpreters."

"... a programmer's dream."

"... wonderful technical assistance."

"... increased our productivity by a factor of 40."

"... the best C product ever, in any category."

- **Price: $300.00 (Demo $45.00) MC, VISA**

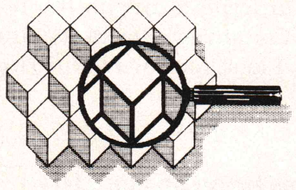
Prices include documentation and shipping within U.S. PA residents add 6% sales tax. Specify compiler.

- C–terp runs on the IBM PC (or any BIOS compatible machine) under DOS 2.x and up with a suggested minimum of 256 Kb of memory. It can use all the memory available.

* C–terp is a trademark of Gimpel Software.

GIMPEL SOFTWARE
3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

OF INTEREST



PCC/Systems has introduced cc:Mail for local-area networks. With this system, users of the IBM PC and compatibles can send, receive, store, and edit electronic message envelopes. Anything created on a PC, such as text, graphics, data files, and display screens from other software application programs, can be sent to another PC in an envelope. The cc:Mail package includes a full-featured word processor with a built-in, felt-tip highlighting pen and a graphics/drawing package. Users have a choice of color palettes and can edit materials from other sources. Screen content from any application program can be frozen to create snapshots, which can then be edited and inserted into messages. Each message can include any combination of up to 20 text, graphics, and file items. The starter package for ten users costs $995, and expansion kits are available.

Ashton-Tate has introduced the dBASE III Plus LAN Pack, designed to enable multiple users to share dBASE Plus files on a local-area network. Equipped with built-in multiuser and stand-alone capability, dBASE III Plus can network an unlimited number of users. Combining one dBASE III Plus with one LAN Pack provides a four-user network. dBASE III Plus and dBASE III Plus LAN Pack can run on any network that supports PC-DOS, Version 3.1, including the IBM PC Network, 3Com's 3Plus, and Novell's Advanced NetWare, Version 1.01, or later. Other LANs that use an operating system compatible with PC-DOS 3.1 can also be supported.

Kyss!, an advanced operator interface with system enhancement facilities, is available from **The Information People.** Kyss! includes a menu generator, extended DOS batch-processing capabilities, disk file management, job tracking, and a text display facility for preparing on-line instructions. The suggested retail price is $150 for single-user versions; network versions begin at $300 for six-user systems.

The MLink Data Communications System from **Corporate Microsystems** includes the Development System, a software package recommended for system developers. MLink also includes terminal emulation, an on-line help system, a configuration system, three file-transfer protocols, a script interpreter, a script debugger, a script assembler, 24 application scripts, and script source files. The system is designed to facilitate micro-to-mainframe, micro-to-micro, PC-to-Unix, and other data-communication links. Prices vary according to configuration.

Information Technologies' LinkUp 3270 SNA, 3270 BSC, and 3770 SNA emulations have been enhanced. The products feature foreground/background operations, enhanced multiple printing capacities, and up to 32 simultaneous sessions. They support printer output to standard DOS printers, a high-speed serial printer, and spooling of print files to disk. All modes can be driven simultaneously. The stand-alone coprocessor version is available for $995.

VersaLAN is a local-area network solution from **Software Clearing House** that includes software, hardware, and wiring. VersaLAN can handle up to 32 PCs and is compatible with PC-DOS and MS-DOS and the IBM PC Network. It also has software hooks so that advanced users can program it to add functions such as private data encryption or links to user-written software. In addition, VersaLAN features electronic mail, file transfer between micros, and hard disk and printer sharing. The product costs $250 per PC; additional PC connections cost from $175.

Polygon Associates has enhanced its poly-COM/240 terminal-emulation and file-transfer communication software package. The package now offers hot-key control for switching between a DOS screen and VAX terminal session without losing the communication link, instant toggle between text and graphics modes, host control commands for unattended or distributed applications, and a screen-saver function that blanks the PC screen after a defined period to reduce video screen wear.

Security

Several products that guard against unauthorized computer access are available from **Digital Pathways.** The Defender IID provides direct-dial user verification. The verification process takes only one phone call, requiring a valid access code, password, and/or SecureNet Key validation. The Defender IIk is a data encryption manager for highly secure dial-up links. It enables remote users with PCs to install an encryption board to protect data in transmission. The SecureNet Key provides an additional level of security for any Defender II system. Each user is provided with a credit-card-size key that has been initialized with a unique key number. The user then arms the key with a personal identification number. Prices vary according to configuration.

Release 2.0 of **System Automation Software's** Logger computer resource monitor is a RAM-resident program that tracks and documents the everyday use of IBM PC, PC/XT, PC/AT, and compatible computers. The new version includes log-in security and a summary option in the reporting subsystem that summarizes computer usage by user, directory, and program and calculates the duration of each activity. Logger's retail price is $74.95.

Artificial Intelligence

XSYS is an advisory expert-system shell from **California Intelligence** that runs on IBM PCs. In a typical scenario, the system asks the user problem-related questions and displays selection menus, depending on which specialized knowledge base the user has selected to attach to the generic XSYS shell program. The system can also explain, step-by-step, its pro-

gress and conclusions during and after each consultation. XSYS' facilities include knowledge attributes, variables, and operators in the *if* and *then* parts of any rule, handling of uncertainty and negation, and automatic concatenation of hierarchically related knowledge bases. The system requires an IBM PC, PC/XT, or PC/AT with at least 640K and DOS 2.X. The license fee for a single-CPU copy is $995.

Borland International's fifth-generation language development system, Turbo Prolog, is designed to infer or derive information from stated facts. The PROLOG language employs a theorem-proving algorithm for logic programming in order to take a set of premises and arrive at an appropriate conclusion. The algorithm utilizes pattern matching and back tracking. Turbo Prolog is priced at $99.95 and is available for IBM PC and compatible microcomputers.

The GCLisp 286 Developer from **Gold Hill Computers** includes a memory interpreter that features lexical scoping and the ability to address up to 15 megabytes of physical memory. It also features a memory compiler designed to allow applications to run up to 15 times faster than normal. The GCLisp 286 requires an IBM PC/AT or compatible with at least 2 megabytes; PC-DOS 3.0 or later; one double-sided, double-density or quad-disk drive; and a hard disk.

Application Development

Orchid Technology's PCturbo 286e utilizes an 8-MHz 80286 CPU, 80287 match processor socket, and a 16-bit internal system bus. With the connection of an optional RAM card, the PCturbo 286e accommodates the Lotus/Intel/Microsoft Expanded Memory Specification. PCturbo 286e with 1 megabyte of RAM costs $1,195.

MasterSweep-File Maintenance Utility is a disk utility from **The Software Store** that gives users access to disk directories and files. It enables users to find, view, copy, print, rename, move, delete, or tag files, and it supports path directories under PC-DOS. MasterSweep costs $49 and is available for PC-DOS and CP/M-80 computers.

PRO-C is a software package from **Majic Software** that steps a developer through an application definition and then generates a C program. The product comes with a half-megabyte of context-sensitive, on-line help that is available at the touch of a key. Its generic record-retrieval mechanism allows users to interface a generated program with an existing file system. Alternatively, users can select the interface to industry products such as Btrieve and C-ISAM. The C compilers currently supported by PRO-C are Microsoft's Version 2.0, Lattice's Version 2.15, Computer Innovations' Version 2.3F, and DeSmet's Version 2.4. PRO-C costs $195.

Fort's Software has announced V-EMM, the Virtual Expanded Memory Manager. V-EMM provides up to 8 megabytes of virtual expanded memory and can execute many unaltered programs that support the Lotus/Intel/Microsoft Expanded Memory Specification. The price is $89.95.

Calvert Computer Systems has released Professional Applications System (PAS) software for program developers. PAS consists of four programs: two for generating applications

ASMLIB

Assembly Language Programming Library for the IBM PC/XT/AT or compatible DOS systems.

ASMLIB gives the Assembly Language programmer 190+ assembly functions which do -

- Graphics functions draw CIRCLES, ARCS, ELLIPSES, LINES, and plots POINTS on the Color Adapter, Enhanced Graphic Adapter, and the Herc. Monochrome Card. Functions also allow PAINTING, IMAGE SAVES and RESTORES, and SCROLLING.
- Text Windows - Up to 64 text windows may be defined, outlined, overlapped, moved, and can be grouped onto 256 logical display pages.
- Floating Point - Arithmetic and Trigonometry functions for the MS and IEEE (8087) floating point formats (both 4 and 8 byte precision), and the 8087 (80287) can be utilized automatically if installed into the target system.
- ASCII String/Numeric conversion routines provide a user interface to the math functions. ASFORMAT function allows numeric values to be formatted utilizing commas, dollar signs, left or right justified, etc. (i.e. BASIC's PRINT USING).
- Mouse Support - ASMLIB provides support for any mouse device which adheres to the MS Mouse software standard.
- Dynamic Memory support can utilize all available memory (up to 640k). Blocks of memory can be allocated, changed, moved, and killed.
- Console I/O, Disk I/O with file copy routines, Asynchronous Communications, Printer Support, ASCII String support, Sound generation, plus much more.

ASMLIB is supplied with complete MASM source code on 3 DOS diskettes along with a 215+ page reference manual.

- ALL FOR ONLY -
$149.00

For ordering or info please call:

BC Associates
13073 Springdale St., Suite 134
Westminster, CA 92683
(714) 741-3015

Phone COD orders accepted - ORDER YOURS TODAY!!!

Circle **no. 182** on reader service card.

M68000 SINGLE BOARD COMPUTER

On board 6-10 MHz CPU, 20K RAM, 32K EPROM, two RS-232, 16-bit port, 5-counter/timers expandable via Memory/FDC Board.

M68K CPU (bare board)	$ 89.95
M68K CPU A&T (6MHz)	$495.00
MD512K Memory/FDC (bare board)	$ 89.95
MD512K Memory/FDC (128K)	$495.00
FDC/Hard Disk interface option	$150.00
M68KE Enclosure w/power supply	$249.00
M68K Monitor EPROM's	$ 95.00
M68K Macro Cross Assembler	$195.00
4XFORTH OS w/assembler, editor	$295.00
CP/M 68K OS w/''C'' compiler	$395.00

EMS Educational Microcomputer Systems
P.O. Box 16115
Irvine, CA 92713
(714) 854-8545

Circle **no. 103** on reader service card.

(continued from page 121)

and two for running them. Programs are menu-driven and support full-screen editing, color systems, function and arrow keys, and unlimited nesting of menus. PAS applications can run any .COM or .EXE program. The system requirements are MS-DOS, 256K, two 360K disk drives, and an ANSI.SYS driver. The product costs $49.95.

Specialized Systems Consultants has announced an IBM PC, PC/AT, or Xenix System V port for its Unix-based Hitachi 6301 C Cross Compiler. The port features a stack frame designed to minimize overhead, separate compilation and linking, a source-code optimizer, and a run-time library.

The Synergy Development Toolkit from **Matrix Software Technology Corp.** provides tools for accessing Synergy run-time function calls from a wide range of languages. These language gateways are designed for the IBM/MS Macro Assembler, Turbo Pascal, IBM/MS BASIC Compiler and Pascal Compiler, Lattice C Compiler, IBM/MS BASIC Interpreter, Microsoft C Compiler, and dBASE II/III. The toolkit features a font collection, compiler, manager, and graphics resource editor. The retail price is $395.

Microtec Research has introduced the ASM180 cross assembler package, a full implementation of a relocatable macro assembler for the Hitachi HD64180 microprocessor's specified assembly language. ASM180 is available on DEC VAX/VMS and MicroVAX/VMS. The assembler features a compatible instruction set and directives, conditional assembly, symbolic addressing, relative addressing, and symbol cross-reference listing. A binary license for the ASM180 is $2,000 for the VAX under VMS or Ultrix.

The following utilities are available from **Lattice**: Lattice Text, a collection of eight programs that provide a set of tools for examining and editing text files; Lattice Make, an automated product generator; Lattice Screen, which provides error tracking; and Lattice dBC III Library, which contains more than 70 C functions.

BlueFish, a software product from **Computer Access Corp.**, provides full text-management capability for users of IBM PCs and compatibles. BlueFish can handle correspondence, contracts, medical journals, government regulations, engineering specifications, legal research, insurance documents, or company personnel records. The software operates on a minimum configuration of a PC with 256K, one disk drive, and PC-DOS. The package comes in two configurations: an office-automation, full-test, management system with both build and search/retrieval modules; and a search and retrieval module designed for publishers of data distributed on optical or other mass-storage media. Site licenses start at $750 per site.

Languages

PforCe, a library of object-oriented functions and subsystems for C, is available from **Phoenix Computer Products Corp.** Written in C and assembly language, PforCe offers programmers both high- and low-level functions that are ready to use. High-level functions allow programmers to manipulate windows, screens of fields and Lotus-like menus, and databases as objects. Low-level functions give programmers hardware control and enable them to change defaults at will. Sophisticated subsystems are also offered. PforCe is available for Microsoft, Lattice, Computer Innovations, and Wizard compilers. It is priced at $395.

The Greenleaf Comm Library (Version 2.0), a programmers' tool supporting C, is available from **Greenleaf Software.** More than 120 functions are provided to support communications at up to 9,600 baud, up to 16 simultaneous channels, XON/XOFF and XMODEM protocols, and flow-control options. Version 2.0 supports PC-DOS and MS-DOS and retails for $185.

Absoft Corp. has added two members to the MacFortran family: MacFortran/020 and MacFortran/RL. Both were designed to take advantage of new high-performance Macintosh hardware, General Computer's HyperDrive 2000, and Levco's MAC Super 20FP. MacFortran/020 is designed to generate code for Macintoshes upgraded with an MC68020 CPU and MC68881 math coprocessor. MacFortran/RL is a series of replacement run-time libraries for MacFortran users using hardware floating point. It supports NS32081 boards, the MC68881, or General Computer's HyperDrive 2000.

FORTRAN-80 Utilities from **Cleydale Engineering** are designed to complement the Microsoft FORTRAN-80 compiler, which runs under CP/M-80. These utilities consist of an optimized, scientific, subroutine library; Forlib.Rel math addition; an escape sequence and control character generator for controlling peripheral devices; and three FORTRAN programming tools. Each subroutine module is furnished with a demonstration driver program. Subject areas include linear and nonlinear regression analysis for curve fitting experimental data, statistics, matrix operations, simultaneous equations, forward and inverse fast Fourier transforms, numeric integration, equation roots, and graphics. The entire package consists of 60 files occupying 238K of disk storage and is distributed in the standard single-sided, single-density CP/M format. The cost is $49.95.

TDI Software has released the UCSD Pascal compiler, which includes the p-System operating system, for the Atari 520ST. UCSD Pascal features support units for separate compilation, assembly-language procedures, full implementation of Pascal, and a full-screen editor. The product is available in a regular and a developers' version. In addition to the UCSD Pascal compiler and p-System operating system, the developers' version contains an M68000 assembler, a native code generator, a symbolic debugger, and assorted Pascal units for manipulating directories and performing systems work. UCSD Pascal for the Atari 520ST is priced at $79.95; the developers' version is priced at $149.95.

Philon's Fast/Pascal is a high-quality compiler developed for use by programmers in the scientific and educational communities. It is designed for the 16- and 32-bit environments. Programs written in Fast/Pascal are portable to other hardware/operating systems. It is fully compatible with IEEE standard floating-point real arithmetic, and system calls are executable from within

the language. The package also contains a set of run-time libraries, file-handling routines, and string-handling capabilities.

MasterForth, which is available from **Micromotion**, is an implementation of the Forth programming language that includes a 68000 macro assembler and a full interface to CP/M 68K. Relocatable utilities and transient definitions make it possible to run software packages, and a string package, screen editor, and resident debugger are standard features. MasterForth is also available for the Macintosh, the IBM PC family, the Apple II family, the Commodore 64, and CP/M Z80 machines. It retails for $125.

For IBM/Apple

Release 1.2 of TextBank from **Group L Corp.** offers improved performance through the use of extended memory, support for additional storage devices, several extensions to the search and user interface, profiles of the most widely used Dialog and BRS on-line databases to make information searchable when downloaded, and full support for individual text files of up to 20 megabytes. TextBank requires an IBM PC or compatible with 640K, a hard disk, and MS-DOS. It is available for $995.

The Polytron Version Control System (PVCS) from **Polytron Corp.** is a source-code version and revision management system for programmers or teams of programmers developing large or complex programs on PCs and networks. The system maintains the revision history of source files and chronological and historical records of changes. It reconstructs any prior revision of any module, defines a version as specified revisions of various modules, and supports branching from prior revisions. Disk space is conserved by an intelligent difference detector that stores only the difference between successive revisions of a module. A single-user licence is available for $395.

Show Partner, a memory-resident graphics editor with animation, is available from **The Marketing Channel**. One image can quickly and completely replace another, and wipe, split, and box effects perform their transitions along traveling event line edges moving up or down, left or right, together or apart, or in or out. Show Partner also offers a scroll effect in any of four directions, fade-in of a selected area, and two-part weave of an area. The program loads into RAM alongside other applications and can also work in a nonresident stand-alone mode. Text or graphic screens are captured from any source and saved as named files. The program has the ability to add or change colors, size rotate, move graphics, and add text to graphics. Show Partner supports IBM CGA and EGA displays as well as the AST ColorGraphPlus palette display, AST Preview, or Hercules-compatible monochrome graphics adapters. It costs $149.

Version 3 of **Portable Software**'s PortaAPL software package for the Macintosh is a full-featured interpreter for standard APL. PortaAPL features a full-screen editor, the ASCII character set option, and the Host File System option. Also, system functions are available for accessing many of the Macintosh toolbox ROM routines, such as QuickDraw graphics, communications, sound generation, and menus.

FTL Modula-II
$49.95!

WORKMAN & ASSOCIATES USER-FRIENDLY SOFTWARE

Your next computer language. The successor to Pascal, Modula is powerful. Why? Once a routine is written, it need never be recompiled. Programs work everywhere from Z80 through VAX.

FTL Modula-II is a full compiler for Z80 CP/M and MSDOS computers! It's **fast** -- 18K source compiles in 7 seconds! The built-in split screen editor is worth $60 alone. Some standard features: full recursion, 15 digit reals, CP/M calls, coprocesses, assembler and linker. The one-pass compiler makes true .COM, ROMable code, too. Get the language you've waited for now. Only $49.95!

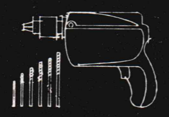
FTL Editor Toolkit

Full source to our split-screen programming editor. Curious? Want to customize to your tastes? Want sample Modula-II code? This is perfect for you. Comes with all you need for your personal editor or terminal installer. Just $39.95!

Workman and Associates
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

We have over 200 formats in stock! Please specify your format when ordering. Add $2.50 per order for shipping. We welcome COD orders!

Circle **no. 244** on reader service card.



Power Tools for system builders™

Call today for our free catalog of design aids, compilers, libraries, debuggers, and support tools for Apple and IBM micro computers. The Power Tools catalog includes product descriptions, warranty and license terms, and all the information you need to make an intelligent purchase decision.

TSF offers technical support, competitive pricing, free UPS shipping on orders over $100, and a reasonable return policy. Visa, MasterCard, and American Express accepted without surcharge. **TSF helps you get your job done.**

Sample Prices:
Microsoft C $259
MASM 4.0 $109
Turbo Pascal $45
Mark Williams C $375
Lets C $59
Wendin OS Toolbox $89
Blaise Async Manager $137

Call Toll Free
24 hrs a day/7 days a week
Ask For Operator 2053
800-543-6277
Calif: 800-368-7600

TSF *The Software Family™*
• Dept. C-2 • 649 Mission Street
• San Francisco • CA 94105
• (415) 957-0111

Circle **no. 230** on reader service card.

OF INTEREST
(continued from page 123)

The package for the Macintosh is priced at $275. Current customers can upgrade to Version 3 for a $25 fee.

Reference Map

Absoft Corp., 4268 N. Woodward, Royal Oak, MI 48072; (313) 549-7111. Reader Service Number 16.
Ashton-Tate, 20101 Hamilton Ave., Torrance, CA 90502-1319; (213) 329-8000. Reader Service Number 17.
Borland International Inc., 4585 Scotts Valley Dr., Scotts Valley, CA 95066; (408) 438-8400. Reader Service Number 18.
California Intelligence, 912 Powell St., #8, San Francisco, CA 94108; (415) 391-4846. Reader Service Number 19.
Calvert Computer Systems Inc., 240 E. Main St., P.O. Box 95, Athena, OR 97813; (503) 566-3338. Reader Service Number 20.
Cleydale Engineering, Rte. 1, P.O. Box 217-B, Blacksburg, VA 24060. Reader Service Number 21.
Computer Access Corp., Ste. 324, 26 Brighton St., Belmont, MA 02178-4008; (617) 484-2412. Reader Service Number 22.
Corporate Microsystems Inc., P.O. Box 277, Etna, NH 03750; (603) 448-5193. Reader Service Number 23.
Digital Pathways Inc., 201 Ravendale Dr., Mountain View, CA 94043; (415) 964-0707. Reader Service Number 24.
Fort's Software, P.O. Box 396, Manhattan, KS 66502; (913) 537-2897. Reader Service Number 25.
Gold Hill Computers Inc., 163 Harvard St., Cambridge, MA 02139; (617) 492-2071. Reader Service Number 26.
Greenleaf Software Inc., 1411 LeMay Dr., Ste. 101, Carrollton, TX 75007; (214) 446-8641. Reader Service Number 27.
Group L Corp., 481 Carlisle Dr., Herndon, VA 22070; (703) 471-0300. Reader Service Number 28.
Information People (The), 443 Hudson Ave., Newark, OH 43055; (614) 349-8644. Reader Service Number 29.
Information Technologies Inc., 7850 E. Evans Rd., Scottsdale, AZ 85260; (602) 998-1033. Reader Service Number 30.
Lattice Inc., P.O. Box 3072, Glen Ellyn, IL 60138; (312) 858-7950. Reader Service Number 31.
Majic Software Inc., 224 S. Salisbury, Ste. C4, West Lafayette, IN 47906; (317) 743-0610. Reader Service Number 32.
Marketing Channel Ltd. (The), 120 E. Washington St., Ste. 421, Syracuse, NY 13202; (315) 474-3400. Reader Service Number 33.
Matrix Software Technology Corp., 50 Milk St., 15th Floor, Boston, MA 02109; (617) 723-8327. Reader Service Number 34.
Micromotion, 8726 S. Sepulveda Blvd., #A171, Los Angeles, CA 90045; (213) 821-4340. Reader Service Number 35.
Microtec Research, 3930 Freedom Cir., Santa Clara, CA 95054; (800) 554-5554, in CA (408) 733-2919. Reader Service Number 36.
Orchid Technology, 47790 Westinghouse Dr., Fremont, CA 95439; (415) 490-8586. Reader Service Number 37.
PCC/Systems, 480 California Ave., Ste. 201, Palo Alto, CA 94306; (415) 321-0430. Reader Service Number 38.
Philon Inc., 641 Avenue of the Americas, New York, NY 10011; (212) 807-0303. Reader Service Number 39.
Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062; (617) 762-5030. Reader Service Number 40.
Polygon Associates Inc., 1024 Executive Pkwy., St. Louis, MO 63141; (314) 576-7709. Reader Service Number 41.
Polytron Corp., P.O. Box 787, Hillsboro, OR 97123; (503) 648-8595. Reader Service Number 42.
Portable Software, 60 Aberdeen Ave., Cambridge, MA 02138; (617) 547-2918. Reader Service Number 43.
Software Clearing House Inc., 771 Neeb Rd., Cincinnati, OH 45238; (513) 451-6742. Reader Service Number 44.
Software Store (The), 706 Chippewa Sq., Marquette, MI 49855; (906) 228-7622. Reader Service Number 45.
Specialized Systems Consultants Inc., P.O. Box 55549, Seattle, WA 98155; (206) 367-UNIX. Reader Service Number 46.
System Automation Software Inc., 8555 16th St., Silver Spring, MD 20910; (301) 565-8080. Reader Service Number 47.
TDI Software Inc., 10410 Markison Rd., Dallas, TX 75238; (214) 340-4942. Reader Service Number 48.

—Wendelin Colby

DDJ

Circle **no. 219** on reader service card.

Now You Know Why BRIEF™ is BEST

"BRIEF, The Programmer's Editor, is simply the best text editor you can buy." John Dvorak, INFOWORLD 7/8/85

"A bona fide Undo..."

Steve McMahon, BYTE 3/85

As Mark Edwards describes in DR. DOBB'S JOURNAL (11/85), "BRIEF has an outstanding undo facility. The default configuration allows the last 30 editing commands to be undone. This number can be raised to a maximum of 300 commands. Until you reach this maximum or run out of RAM, every command you issue can be undone. So if you make ten changes and then realize that the first one was an error, you can undo all the changes back to the mistake... Needless to emphasize, this facility can save endless grief."

No other editor has this capability.

The Program Editor with the BEST Features

Since its introduction, BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features **MOST ASKED FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined, including the ability to configure windows, keyboard assignments, and commands to **YOUR** preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, "(BRIEF)...is quite simply the best code editor I have seen."

Every Feature You Can Imagine

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support, like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)
- Unlimited File Size -(even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

Program Editing YOUR Way

A typical program editor requires you to adjust your style of programming to its particular requirements – NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key – even basic function keys such as cursor-control keys or the return key.

The Experts Agree

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion – BRIEF IS BEST!

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!

NOT COPY PROTECTED

Solution Systems™

MONEY-BACK GUARANTEE
Try BRIEF ($195) for 30 days – If not satisfied get a full refund.
TO ORDER CALL (800-821-2492)

SOLUTION SYSTEMS, 335-P WASHINGTON ST., NORWELL, MA 02061, 617-659-1571

BRIEF is a trademark of UnderWare

Circle **no. 147** on reader service card.

uniForth™

NOW: a FORTH System that meets *your* needs!

Available for most personal computers and DEC minis, this FORTH-83 language from an established company will meet the needs of the *novice* and the *expert* alike!

Beginners will enjoy the clear style of the Users Guide. The video editor is easy to use, and the debug/decompile utilities make programming a snap. Leave questions on our Bulletin Board Service for quick answers. Our optional telecommunications package provides superb modem access.

Advanced Programmers will get more details of the system from the comprehensive Programmers Guide. Optimized code and full memory space utilization make **UNIFORTH** one of the fastest FORTHs for large applications. Use the complete macro assembler for time-critical routines. Text file, operating system, graphics, sound, and string support are included.

Engineers will use the multitasking and high-level interrupt features to handle simultaneous functions. We have ROM-based systems for many popular single-board computers, or you can use one of our optional cross-compilers for one of a dozen processors to generate your own systems. Need help? We offer custom programming, classes and other consulting services.

Scientists will appreciate the full floating-point support. Optional utilities include a 2-D plotting package, astronomical programs and catalogs, image processing and artificial intelligence.

Public-Domain Samplers are available for many systems.

Call or write for our **free catalog.** We guarantee that you'll enjoy UNIFORTH, or your **money back**!

Unified Software Systems
P.O. Box 21294
Columbus, OH 43221
(614) 459-7735
BBS: (614) 459-7736
(300/1200 baud, 24 hrs.)

Circle **no. 188** on reader service card.

ADVERTISER INDEX

*This advertiser prefers to be contacted directly: see ad for phone number.

ADVERTISING SALES OFFICES

Southeast
Gary George (404) 355-4128

Northern California/Northwest
Lisa Boudreau (415) 366-3600

Midwest
Michele Beaty (317) 875-8093

Southern California/AZ/NM/TX
Michael Wiener (415) 366-3600

Advertising Director
Robert Horton (415) 366-3600

The Peak of Performance

SCALE THE HEIGHTS OF PRODUCTIVITY

Sure, you've proven that in your hands a computer is a productive tool. But if you haven't teamed up with a SemiDisk you have heights yet to climb!

IT'S NO MERE RAMDISK

SemiDisk has been leading the way for Disk Emulators since their inception. If you've seen RAMdisks you know what it's like to load programs in an instant, and read or write files without delay. Unlike alternatives, the SemiDisk offers up to 8 megabytes of instant-access storage while leaving your computer's main memory free for what it does best - computing!

KEEP A GRIP ON DATA

Go ahead, turn off your computer. Take a vacation. With the battery backup option, your valuable data will be there in the morning even if you aren't. You'll sleep better knowing not even a 5 hour blackout will sabotage your files.

NEW LOWER SEMIDISK PRICES THAT WON'T SNOW YOU UNDER

	512K	2Mbyte
IBM PC, XT, AT	$495	$995
Epson QX-10	$595	$995
S-100, SemiDisk II	$799	$1295
S-100, SemiDisk I	$595	——
TRS-80 II, 12, 16	$695	$1295
Battery Backup Unit	$130	$130

Software drivers available for CP/M 80, MS-DOS, ZDOS, TurboDOS, and VALDOCS 2.

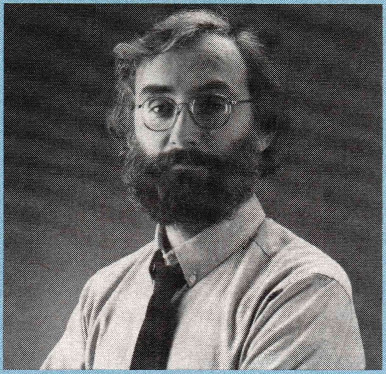
SEMIDISK

SemiDisk Systems, Inc.
P.O. Box GG, Beaverton, Oregon 97075
503-626-3104

Call 503-646-5510 for CBBS/NW, and 503-649-8327 for CBBS/Aloha, all SemiDisk equipped computer bulletin boards, (300/1200/2400 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle **no. 85** on reader service card.

SWAINE'S FLAMES



The declaration that some British colonists on this continent made of their independence 200-odd years ago this month was not the first rebellion against a regime regarded as tyrannical, of course; nor was the government they later formed the first democracy. Nothing ever happens for the first time.

The authors of the Declaration of Independence were particularly eloquent in their defense of the rights of the nonindentured adult white male, and over the succeeding centuries various of their descendants have been eloquent about the need to extend recognition of those rights to all Americans. Others have been equally eloquent about the need to deny such rights to blacks, women, homosexuals, and people of suspect ancestry in time of war. Americans are often eloquent on the subject of independence, even if a little afraid of it in practice. Just think what they'd be like if they were as independent as they are eloquent. Why, they'd be like Forth programmers.

Forth programmers seem to be a particularly independent bunch. The common explanation, which is probably correct, is that the language encourages independent thinking. If so, it fully justifies this special Forth issue, our sixth. Events in the Forth community are no less interesting now than in 1981 when we started this annual tradition. In last year's Forth issue, you may recall, Leo Brodie led a tour of the architecture of the then-new Novix Forth chip, a structure for which Forth creator Charles Moore did the blueprints. Now Harris Semiconductor has acquired a Novix license and is supplying bit-slice versions of the design. What this added support could mean to Novix and to Harris will be determined by the Forth programming community. Predictions are foolhardy.

It had been exactly two years, I remarked last month in this space, since we had reviewed Borland's Turbo Pascal, and now here was Turbo Prolog knocking at the door. Well, not exactly at the door, but I did finally manage to get my beta and final versions of the new Turbo Prolog without driving to Scotts Valley and have since been comparing notes on the product with Juergen Fey, an editor with *PC Magazin*. (*PC Magazin* is the magazine that Markt & Technik, M&T Publishing's Prussian parent publisher, puts out for the fairly technical PC readership in West Germany, and Juergen is one of its more technical editors.)

Turbo Prolog is PROLOG, Juergen thinks, in the same sense that Turbo Pascal is Pascal. The user interface is excellent, the performance is fast, and there are many extensions that overcome deficiencies of pure PROLOG. The result—our preliminary impression only—is that Turbo Prolog is at least a very good environment for learning Turbo Prolog. Beyond that—well, we are planning to review the product in the near future, and we'll evaluate it as a language implementation for first-time users, as a serious development environment, and as a PROLOG implementation. We'll consider what its existence, price, and ease of use could mean for the spread of PROLOG, and we'll evaluate its speed in light of Borland's claims and determine what the programmers gave up for the speed they got.

Borland has taken some brave steps in the past; I wonder if its independent spirit will be affected if the company goes public this year, as seems likely.

As an editor and writer I count the freedom to read among those rights alluded to at the top of this page, and I would not be an editor if I failed to bring to your attention the cloud of censorship that is once again moving over this country. It's starting with "adult" (i.e., adolescent) entertainment, with guidelines on the treatment of sex in films and the removal of *Playboy* et al. from convenience stores. *DDJ* isn't in the prurient interest business, of course, but we are currently running an on-line conference on the politically charged issue of data encryption, and I doubt that the cloud will stay over someone else's backyard for long.

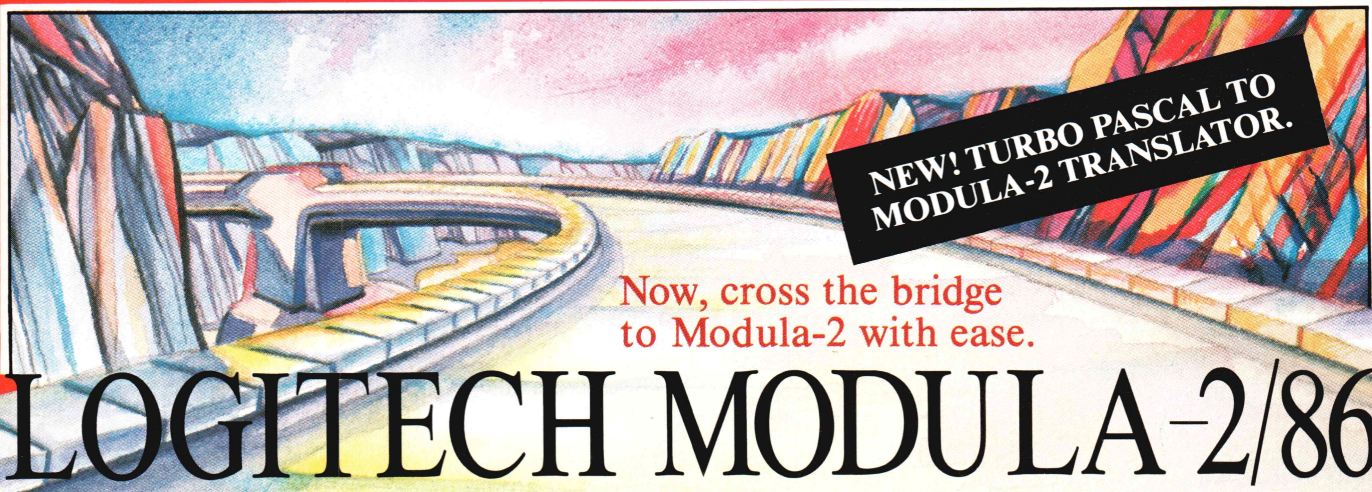
Ray Borrill, a pioneering computer retailer, recently flamed to me about the wild crowd at the Midwest Computer Show in 1976, where he shared a booth with Bob Marsh and Steve Dompier of Processor Technology. It reminded me of the Atari and Commodore booths this year in Atlanta, packed with independent third-party developers. In the otherwise stuffy Comdex atmosphere, the air in those booths was almost giddy—almost evocative of 1976.

Desktop publishing was the big thing at Comdex this year, with laser printers and document scanners coming down in price. Phoenix held a meeting for 80386 developers to discuss setting 80386 standards before IBM does it. Despite a lot of talk that copy protection is going the way of King George, the Tories of Lotus seem unconcerned; meanwhile, Central Point Software, purveyor of guns to the rebels, is looking into supplying armor to the redcoats.

Michael Swaine

Michael Swaine
editor-in-chief

This is the Modula-2 compiler *everybody's* been waiting for...



LOGITECH MODULA-2/86

This is Modula-2 at its *absolute* best. It's a fully integrated development environment that takes into account what you need as a programmer. Without leaving the Editor, you can call the compiler, linker and utilities.

With Logitech's Modula-2, you'll have the ability to edit several files at once, comparing, window to window, various code modules. You can even move from window to window compiling, linking, debugging and running.

The compiler has the kind of power and room to breathe that you really need in today's complex applications. It is as *easy* to use as Turbo Pascal, without your programs being limited to 64K of code.

At your command will be the libraries of modules that make Modula-2 a programmer's dream. It has essentially the same structure as Pascal with the major addition of a library organization of code modules that allow you to put together programs on a solid, block-by-block, foundation of proven code.

Whether you're working with a module of your own making, or one of the many in our library, you'll find the system by which each module is identified, described and stored an organizational masterpiece. And that's at the heart of Modula-2.

Underneath the sophisticated system is a Modula-2 compiler that is the result of years of development and proven use in industry. We run on the Vax*, and we run on the IBM PC. And the code is portable–from one to the other.

Best of all . . . you can have it right now!

Logitech Modula-2/86 $89 Complete with Editor, Run Time System, Linker, Cursor-positioning debugger, 8087 Software Emulation, BCD module, Logitech's extended library, Utility to generate standard .EXE files, Turbo Pascal (and standard Pascal, too) to Modula-2 translator *(included without charge until 8/1/86)*, and much, much more!

Logitech Modula-2/86 with 8087 support $129 Even if you haven't yet gotten an 8087 co-processor, you can still use this version.

Logitech Modula-2/86 Plus $189 For machines with 512K or more. Takes advantage of the larger memory to increase compilation speed by 50%! Supports 80186 and 80286 as well as 8086 and 8088. Includes 8087 and 80287 support, too.

Window Package $49 Now you can build true windowing into your Modula-2/86 code with ease, too. Very powerful and very full, yet only 15K in size. Features virtual screens, color support, overlapping windows and a variety of borders.

Run Time Debugger (source level) $69 Much more powerful than just a symbolic RTD. Display source code, data, procedure call chain and raw memory. Set break points, assign values to variables, pinpoint and identify bugs in your source. The ultimate professional's tool!

Utilities Package $49 Features a post-mortem debugger for static debugging. If a program you've written crashes at run time, the situation is frozen, and you can pinpoint, in source, the cause of the error and the data at that moment. Also includes a disassembler, a cross reference utility and a "version" utility that allows conditional compilation.

Make Utility $29 Automatically selects modules affected by code changes for quick and minimal re-compilation and relinking. Even figures out dependencies for you.

Library Sources $99 Source code for our major library modules is now available–for customization or exemplification.

ROM Package If you need to produce rommable code, call our 800 number for further information on this package.

To place an order call our special toll free number

800-231-7717

in California

800-552-8885

Special offer until 8/1/86! **$89** includes
Free! $49.95 value Turbo Pascal translator!
Now, you can take your library with you!

Yes, I'd like to take the next logical step in programming.

Please send my copy of Logitech Modula-2/86 to the following address:

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number ________ Expiration Date ________

Signature ________

Name ________

Address ________

City ________

State ______ Zip ______ Phone (____) ________

Here's the configuration I'd like:

- ☐ Logitech Modula-2/86 $89
- ☐ Logitech Modula-2/86 $129 with 8087 support
- ☐ Logitech Modula-2/86 Plus $189

Please add $6.50 for shipping and handling.

And include the indicated items:

- ☐ Window Package $49
- ☐ Run Time Debugger $69 (source level)
- ☐ Utilities Package $49
- ☐ Make Utility $29
- ☐ Library Sources $99

Total enclosed $ ________

(California residents, please add applicable sales tax)

LOGITECH

LOGITECH, Inc.
805 Veterans Boulevard
Redwood City, California 94063
Telephone (415) 365-9852

For European pricing, please contact:

LOGITECH SA
Box 32, CH-1143 Apples, Switzerland
Telephone 41 (21) 774545

Please call our 800 line for: ☐ Information on our *VAX version ☐ Site License and University Discounts ☐ Dealer and Distributor information

*Turbo Pascal is a registered trademark of Borland International

HAUPPAUGE

Is Getting A Fast Reputation.

HAUPPAUGE started earning a fast reputation with their 87 Math Pak, the combination of an 87 chip and 87 Software Pak that's been accelerating PC math since 1982.

Next came their racy 287 FAST/5, a math coprocessor module with its own 5MHz clock, speeding up PC/AT math by 25%. (Pictured above.)

Now, Hauppauge Unveils the 287 FAST/8A...
Our newest math coprocessor for the PC/AT, the 287 FAST/8A moves out at 8MHz—doubling the speed of each floating point math operation. The FAST/8A accelerates AutoCad, 1-2-3, Symphony, Turbo Pascal, Framework and more. The FAST/8A also runs in PC/AT compatibles including the Compaq Deskpro 286, Sperry PC/IT and TI Business-Pro computers.

...And the 87 Software Pak Version 6.0
Designed to steal the heart of programmers, the 87 Software Pak supports IBM's BASIC Compiler 1.0 and 2.0, and Microsoft's QuickBASIC, executing math-intensive programs up to 20 times faster! The 87 Software Pak also performs FFT's and Matrix operations. For example, a PC (or PC/XT) with an 87 Chip and 87 Software Pak can perform a 512-point complex FFT in just 1.1 seconds. What's more, a PC/AT with a FAST/8A inverts a 25 by 25 element matrix in under 1 second.

Circle **no. 274** on reader service card.

HAUPPAUGE Math Coprocessors

287 FAST/8A 8MHz math coprocessor for PC/AT and compatibles....**$379**

287 FAST/5 5MHz math coprocessor for PC/AT and compatibles.......**$249**

287 Chip PC/AT math coprocessor—runs at 4MHz in PC/AT..........**$219**

87 Chip Math coprocessor for IBM PC, PC/XT and compatibles..........**$129**

87-2 Chip Math coprocessor for 8MHz PC compatibles...**$195**

HAUPPAUGE Math Coprocessor Paks

87 Math Pak V.6.0 87 chip and math coprocessor software support for IBM BASIC Compiler 1.0, 2.0 and Microsoft's QuickBASIC. Plus, Matrix and FFT support, one year of free updates, complete source code and "8087 Applications and Programming"..........**$279**

87 Software Pak V.6.0 Math coprocessor software support as in the 87 Math Pak, but without 87 chip..........**$180**
With any Hauppauge math coprocessor..........**$150**

Recalc+ Math coprocessor support for 1-2-3 version 1A..**$ 95**
With any Hauppauge math coprocessor..........**$ 49**

HFT+ Complete Hayes Fourier Transform Package.......**$125**
With any Hauppauge math coprocessor..........**$ 79**

The 287 FAST/8 Doubles Your PC/AT's Math Speed!
Help your PC/AT get a fast reputation with Hauppauge's new 287 FAST/8A. Call today, or contact your local computer dealer to learn more about Hauppauge's racy product line. And ask for "87 Q & A," our free booklet on math coprocessors.

Hauppauge Computer Works, Inc.
358 Veterans Memorial Highway, Suite MSI,
Commack, New York, USA 11725 • 516-360-3827
Available at your local computer dealer

Hauppauge
(Pronounced "Ha-pog")

We acknowledge the following registered trademarks: 1-2-3 and Symphony: Lotus Development Corporation; AutoCad: AutoDesk, Inc.; IBM: PC, PC/AT, PC/XT; Compaq: Deskpro; Sperry: PC/IT; Texas Instruments: Business Pro; Ashton-Tate: Framework.